# tanulmányok 147/1983

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

# PROCEEDINGS OF THE JOINT BULGARIAN-HUNGARIAN WORKSHOP ON "MATHEMATICAL CYBERNETICS AND DATA PROCESSING"

Scientific Station of Sofia University, Giulechica (Bulgaria), May 3-8, 1982

Tanulmányok 147/1983

A kiadásért felelős:

*DR VÁMOS TIBOR*

Editors:
Szerkesztették:

*J. DENEV,*
*B. UHRIN*

Főosztályvezető:

*Demetrovics János*

ISBN 963 311 161 7
ISSN 0324-2951

# CONTENT

DATA BASES AND MATHEMATICAL LINGUISTICS

COMPUTER AIDED DESIGN (CAD) AND PERSONAL COMPUTERS

# F O R E W O R D

*A scientific cooperation has been established between the Mathematical Institute with Computing Centre of the Bulgarian Academy of Sciences /MICC/ and the Computer and Automation Institute of the Hungarian Academy of Sciences /CAI/ in the theme: "Mathematical Cybernetics and Data Processing".*

*In the frame of this cooperation a workshop entitled "Mathematical Cybernetics and Data Bases" has been organized in the Scientific Station of Sofia University, Giulechica /Bulgaria/ in the period of May 3-8, 1982.*

*There were 7 participants from CAI /Division of Computer Science/, 15 participants from MICC /Division of Foundations of Cybernetics and Department of Mathematical Linguistics/ and 3 participants from Blagoevgrad Branch of Sofia University.*

*In the course of the workshop a four-day intensive exchange of scientific ideas took place. There were 21 lectures in 3 sections from the following fields: discrete mathematics, data bases, mathematical linguistics, computer aided design /CAD/ and personal computers.*

*This volume contains final versions of lectures read at the workshop divided into three parts:*

- *Discrete mathematics /papers 1.-10./*
- *Data bases and mathematical linguistics /papers 11.-15./*
- *CAD and personal computers /papers 16.-21./*

*Also 2 round-table discussions has been organized at the workshop about the mathematical cybernetics and data bases, respectively.*

*All participants took active part in discussions, that helped in developing future cooperation.*

*The stay in the Scientific Station has been very pleasent due not only to the good wheather and beautiful mountains surrounding the station but also to the services of the station. All this contributed to the succes of the workshop. Many non-official discussions, walks, excursions, disco-parties, e.t.c. contributed to the succes of the workshop as well. We can state that the participants left the workshop as good friends that is not a negligible thing as regards the future collaboration.*

*For the succesful organization of the workshop we acknowledge the Bulgarian Mathematical Society and Blagoevgrad Branch of Sofia University. Our thanks go to Scientific Secretariat of CAI for making possible to publish this volume.*

LIST OF PARTICIPANTS:

From Bulgaria:

*G. Angelova, M. Aslanski, O.I. Botusharov, K.N. Cimev,*
*J.D. Denev, M. Filipova, I.D. Gjudjenov, E. Juvkova,*
*R. Leseva, I. Nenova, D. Nikolova, K. Manev,*
*R. Mitkov, R. Pavlov, A.L. Petkov, A. Terziev*

From Hungary:

*J. Demetrovics, F. Halász, K. Kovács, T. Lengyel,*
*L. Máté, T. Remzső, B. Uhrin*

# ПРЕДИСЛОВИЕ

Рабочая конференция "Математическая кибернетика и базы данных" состоялась в Болгарии, в горном курорте Гюлечица с 03.05.82 по 08.05.82 г. Она проводилась в рамках двухстороннего сотрудничества между Институтом математики ВЦ БАН и МТА SzTAKI по теме "Математическая кибернетика и обработка данных". В конференции приняли участие 7 сотрудников МТА SzTAKI /Главный отдел вычислительной науки/, 15 из ИМ с ВЦ БАН /сектор Основы кибернетики и теории управления и лаборатория Математическая лингвистика/, и 3 из Благоевградского филиала Софийского университета.

В течении 4 дней состоялся интенсивный обмен научных результатов. Были проведены 4 заседания, на которых было прочитано 21 научных докладов и сообщений по теме сотрудничества. Были представлены 4 раздела этих тем - дискретная математика, базы данных, математическая лингвистика и CAD и персональные компьютеры. Доклады и общая конференция составляют содержание настоящего сборника.

Красивая горная природа, великолепная погода и отличное обслуживание со стороны персонала Научной станции "Гюлечица" при Софийском университете способствовали в немалой степени для успеха конференции. Личные беседы и новые дружеские контакты имели не меньшее значение, чем участие в заседании официальной программы.

Организаторы конференции благодарят:
- Союз математиков в Болгарии за оказанную поддержку;
- Филиал СУ в Благоевграде за предоставленную возможность осмотра этого красивого города и его окресностей;
- руководство и сотрудников МТА SzTAKI за нелегкий труд при подготовке этого сборника.

# DISCRETE MATHEMATICS

# ДИСКРЕТНАЯ МАТЕМАТИКА

# CLONES WITH SHARPLY TRANSITIVE AUTOMORPHISM GROUP

J. Demetrovics - L. Rónyai
Computer and Automation Institute
Hungarian Academy of Sceinces

## Introduction

Let $E_k$ denote the k element set $\{0,1,\ldots k-1\}$, $k \geq 2$. A clone D over $E_k$ is a nonempty set of operations (switching functions) on $E_k$ which contains all projections and closed under forming arbitrary superpositions. From an algebraic point of view D consists of all polynomial functions of an algebra over the base set $E_k$, for example those of the algebra $\langle E_k, D \rangle$. The free spectrum of D is the sequence $s_n(D)$, $n \geq 0$, where $s_n(D)$ is the cardinality of the free algebra with n free generators in the equational class generated by the algebra $\langle E_k, D \rangle$. From the point of view of multiple valued logic $s_n(D)$ is the number of n-ary functions in the clone D.

The free spectrum $s_n(D)$ is an important invariant of the clone D(see Berman [1], [2], Grätzer [6]). In [5] we have investigated the free spectra of clones selfdual to varions types of permutation groups (semiregular, alternating and symmetric groups). The aim of this paper is to compute the free spectrum of clones consisting of all functions selfdual to a sharply transitive permutation group.

Definitions and notation

All permutation groups are considered to act on the set $E_k$. Let $S_k$ and $A_k$ be denote the symmetric and alternating groups on $E_k$, respectively.

Def.: Let $H \leq S_k$ be a permutation group. H is called sharply $\lambda$ - transitive if for any two sequences $a_1,\ldots,a_\lambda$ $a_i \neq a_j$ and $b_1,..,b_\lambda$ , $b_i \neq b_j$ , $1 \leq i \neq j \leq \lambda$ there is exactly one $\pi \in H$ for which $\pi(a_i) = b_i$, $1 \leq i \leq \lambda$ .

Sharply 1-transitive permutation groups are the regular ones. For $\lambda \geq 2$ there are the following possibilities (Blake-Cohen-Deza [3], Nagao [8] ).

$\lambda$ =2 The group of all linear transformations $x \rightarrow ax+b$ on a finite near field.

$\lambda$ =3 The group of all transformations $x \rightarrow (a.x+b)/(c.x+d)$. Here + and / are the corresponding operations of a finite field and $\cdot$ is either the field or a proper near field multiplication.

$\lambda$ =4 The Mathieu group $M_{11}$ with the usual action (k=11).

$\lambda$ =5 The Mathieu group $M_{12}$ with the usual acton (k=12).

$\lambda$ =k-2 The permutation group $A_k$.

$\lambda$=k-1,k The permutation group $S_k$.

Def.: Let $\beta : E_k^n \to E_k$ be a (partial) function and $H \leq S_k$ be a permutation group on $E_k$. $\beta$ is called selfdual to the permutation group H if for $\underline{x}=(x_1,\ldots,x_n) \in E_k^n$ and $\pi \in H$ $\pi\beta(\underline{x}) = \beta(\pi(x_1),\ldots,\pi(x_n))=\beta(\pi(\underline{x}))$ holds whenever $\underline{x}$, $\pi(\underline{x}) \in$ dom $\beta$.

The following lemma concerns with the extension properties of partial selfdual functions.

Lemma 1 ( [4], [5] ). Let $\beta: E_k^n \to E_k$ be a partial function, selfdual to a permutation group $H \leq S_k$. Then there exists an f: $E_k^n \to E_k$ such that

i. dom f = $E_k^n$,

ii. f is selfdual to H,

iii. if $\underline{x} \in$ dom $\beta$ then $f(\underline{x}) = \beta(\underline{x})$.

The extension is unique if and only if dom $\beta$ intersect each H orbit on $E_k^n$. □

For $H \leq S_k$ let

$D_H = \{ f;$ f is an operation on $E_k$ and selfdual to $H \}$

$D_H$ is the clone of all functions selfdual to H, or in other words $D_H$ is the largest clone D over $E_k$ with the property

$$\text{Aut}\left(\langle E_k, D\rangle\right) = H.$$

We want to determine the spectrum $s_n(D_H)$, $n \geq 1$ where $H$ is a sharply $\lambda$-transitive permutation group on $E_k$ for some $\lambda$. From the point of view of multiple valued logic the most important special cases are:

i. $k$ is a prime and $H = \langle \pi \rangle$ where $\pi$ is a cycle of length $k$. In this case $H$ is a regular permutation group and $D_H$ is a maximal clone.

ii. $H = S_k$. Then $D_H$ is the clone of all homogeneous functions.

The results

Our main theorem stated as follows.

Theorem Let $k \geq 2$ and $H \leq S_k$ be a sharply $\lambda$-transitive permutation group. Then for $n \geq 1$, $\lambda \leq k-1$

$$s_n(D_H) = \prod_{i=1}^{\lambda-1} i^{S(n,i)} \cdot k^{S(n,\lambda) + \sum_{j=\lambda+1}^{k} S(n,j)(k-\lambda)\ldots(k-j+1)}$$

Here $S(n,\ell)$ mean Stirling numbers of second kind.

Corollary 1 ([5]). If $k \geq 2$ and $H \leq S_k$ is a regular permutation group then for $n \geq 1$ $s_n(H_D) = k^{k^{n-1}}$

Proof. From our theorem

$$s_n(D_H) = k^{S(n,1) + \sum_{j=2}^{k} S(n,j)(k-1)\ldots(k-j+1)} \qquad (*)$$

Now let us consider the following identity (Lovász [7])

$$\sum_{l=1}^{m} S(m,l)x(x-1)\dots(x-l+1) = x^m$$

If we substitute x=k and m=n and divide both sides by k we obtain the following:

$$S(n,1) + \sum_{l=2}^{n} S(n,l)(k-1)\dots(k-l+1) = k^{n-1}$$

But

$$S(n,l)(k-1)\dots(k-l+1)=0$$

if $l > k$ therefore

$$S(n,1) + \sum_{l=2}^{k} S(n,l)(k-1)\dots(k-l+1) = k^{n-1}$$

and the left side is just the exponent of k in the (*). □

Corollary 2 ([5]). Let $k \geq 2$ and $n \geq 1$. Then

$$s_n(D_{S_k}) = \prod_{i=1}^{k-2} i^{S(n,i)} \cdot k^{S(n,k-1)+S(n,k)}$$

$$s_n(D_{A_k}) = \prod_{i=1}^{k-3} i^{S(n,i)} \cdot k^{S(n,k-2)+2(S(n,k-1)+S(n,k))}$$

□

In order to prove the theorem we need some preparatory lemmas.

Lemma 2. Let $H \leq S_k$ and $X = \{ \underline{x}^1, \dots, \underline{x}^l \}$ be a set of orbit representatives of H on $E_k^n$. (H acts componentwise on $E_k^n$.) Let $H_i$ be denote the stabilizer of $\underline{x}^i$ e.g.

$$H_i = \{ \pi \in H ; \ \pi(\underline{x}^i) = \underline{x}^i \} \qquad 1 \leq i \leq l .$$

Let

$$k_i = |\{ y \in E_k , \ \pi(y) = y \text{ for all } \pi \in H_i \}| \quad 1 \leq i \leq l.$$

Then

$$s_n(D_H) = \prod_{i=1}^{l} k_i$$

Proof.: If $f: E_k^n \to E_k$ then let $\beta_f$ be the restrictrion of f to the set X. Thus $\beta_f : E_k^n \to E_k$ is a partial function. If f is selfdual to H then $\beta_f$ is selfdual as well and conversely, if $\beta$ is a function selfdual to H for which $\operatorname{dom} \beta = X$ then by lemma 1 there is exactly one $f \in D_H$ $\operatorname{dom} f = E_k^n$ with the property $\beta_f = \beta$.

Therefore

$$s_n(D_H) = |\{ \beta : E_k^n \to E_k ; \operatorname{dom} \beta = X, \ \beta \text{ is selfdual to } H \}|$$

We note that for every $\underline{x}^i \in X = \operatorname{dom} \beta$, $\pi(\underline{x}^i) \in X$ if and only if $\pi \in H_i$ and thus, $\beta$ is selfdual to H if and only if for each i, $1 \leq i \leq l$ and $\pi \in H_i$ $\pi\beta(\underline{x}^i) = \beta\pi(\underline{x}^i) = \beta(\underline{x}^i)$ holds. There are $k_i$ possibility to choose the value $\beta(\underline{x}^i)$ and for different values of i we can choose independently.

$$\left|\{\beta : E_k^n \to E_k \;;\; \mathrm{dom}\,\beta = X,\ \beta \text{ is selfdual to } H\}\right| = \prod_{i=1}^{l} k_i$$

and this proves the lemma. □

Def.: Let $\underline{x} = (x_1, \dots, x_n) \in E_k^n$. The pattern of $\underline{x}$ is the partition P of the set $\{1,2,\dots,n\}$ such that $i \equiv j \pmod P$ if an only if $x_i = x_j$. A pattern P is called $l$-pattern $1 \le l \le n$, if it has exactly $l$ classes.

The number of different $l$ patterns is $S(n,l)$. If $\underline{x} \in E_k^n$ and $\pi \in S_k$ then $\underline{x}$ and $\pi(\underline{x})$ have the same pattern. This implies that for $H \le S_k$ the elements of an H-orbit on $E_k^n$ have same pattern. So, we can speak about the pattern of an orbit.

Lemma 3. Let $k \ge 2$, $n \ge 1$ and $H \le S_k$ a sharply $\lambda$-transitive permutation group. The number of H orbits on $E_k^n$ having a fixed $l$-pattern P is 1 if $l \le \lambda$ and $(k-\lambda)(k-\lambda-1)\dots(k-l+1)$ if $\lambda < l \le k$.

Proof. Let $P_1, \dots P_l$ be the classes of the partition P, and $j_i \in P_i$ $1 \le i \le l$. If $l \le \lambda$ and $\underline{x}, \underline{y} \in E_k^n$ have the pattern P then $x_{j_r} \ne x_{j_s}$, $y_{j_r} \ne y_{j_s}$ for $r \ne s$.

The $\lambda$-transitivity of H implies that there is

a $\pi \in H$ for which $\pi(x_{j_i}) = y_{j_i}$ $1 \le i \le \ell$ therefore $\pi(\underline{x}) = \underline{y}$, $\underline{x}$ and $\underline{y}$ belong to the same orbit.

Now let $\lambda < \ell \le k$ and let

$$Y = \{\underline{x} \in E_k^n \text{ the pattern of x is P and } x_i = j-1 \text{ if } i \in P_j \ 1 \le j \le \lambda\}$$

It is clear that $|Y| = (k-\lambda)\dots(k-\ell+1)$.

If $\underline{x} \neq \underline{y} \in Y$ and $\pi \in H$ so that $\pi(\underline{x}) = \underline{y}$ then $\pi(i) = i$ for $0 \le i \le \lambda - 1$. By the sharp transitivity of H, $\pi = id$, $\underline{x} = \underline{y}$. $\lightning$

This implies that $\underline{x}$ and $\underline{y}$ belong to different orbits. By the $\lambda$-transitivity of H for every $\underline{x} \in E_k^n$ having the pattern P there is a $\pi \in H$ such that $\pi(\underline{x}) \in Y$. The above observations show that Y represents all orbits of H and the lemma follows. □

Lemma 4. Let $H \le S_k$ be a sharply $\lambda$-transitive permutation group and $\underline{x} \in E_k^n$ have an $\ell$-pattern P. Let $\lambda < k$ and

$$K = \{\pi \in H, \ \pi(\underline{x}) = \underline{x}\}$$

and

$$k(\underline{x}) = |\{y \in E_k, \ \pi(y) = y \text{ if } \pi \in K\}|$$

Then $k(\underline{x}) = \ell$ if $1 \le \ell < \lambda$ and $k(\underline{x}) = k$ otherwise.

Proof.: First let $1 \le \ell < \lambda$. In this case K is the stabilizer of $\ell < \lambda$ elements of $E_k$ (the components of $\underline{x}$), hence K is transitive on the remaining $k-\ell$ elements

of $E_k$, thus $k(\underline{x}) = \ell$.

If $\ell \geq \lambda$ then by the sharp $\lambda$-transitivity of H, K= id i.e. K fixes all elements of $E_k$, hence $k(\underline{x}) = k$. □

We are ready to prove the theorem.

Proof.: Let X be a set of orbit representatives of H on $E_k^n$.

By the lemma 2:

$$s_n(D_H) = \prod_{\underline{x} \in X} k(\underline{x})$$

Now let

$$X_i = \{ \underline{x} \in X, \ \underline{x} \text{ has an i-pattern} \} \quad 1 \leq i \leq k .$$

By Lemma 3 $|X_i| = S(n,i)$ if $i \leq \lambda$ and

$|X_i| = S(n,i)\,(k-\lambda)\,(k-\lambda-1)\ldots(k-i+1)$ if $\lambda < i \leq k$.

Using these facts and the result of lemma 4 we obtain the following:

$$\prod_{\underline{x} \in X} k(\underline{x}) = \prod_{i=1}^{k} \prod_{\underline{x} \in X_i} k(\underline{x}) = \prod_{i=1}^{\lambda} k(\underline{x})^{S(n,i)} \cdot \prod_{j=\lambda+1}^{k} k(\underline{x})^{S(n,j)(k-\lambda)\ldots(k-j+1)} =$$

$$= \prod_{i=1}^{\lambda-1} i^{S(n,i)} \cdot k^{S(n,k) + \sum_{j=\lambda+1}^{k} S(n,j)(k-\lambda)\ldots(k-j+1)}$$

The proof is complete. □

References

[1] Berman, J., Algebraic properties of k-valued logics, Proc. 10. Int. Symp. on Multiple-valued logic, Evanston Illinois 1980.

[2] Berman, J., Freee spectra of 3-element algebras, Proc.4. Int. Conf.on Universal Algebra and Lattice Theory Puebla Mexico 1982.

[3] Blake, I.F., Cohen, G., Deza, M. Coding with permutations, Information and Control, 43 (1979) 1-19.

[4] Demetrovics, J., Hannák, L., Rónyai, L, Selfdual classes and automorphism groups, preprint 1982.

[5] Demetrovics, J., Rónyai, L., On free spectra of selfdual clones, submitted for publication to the Bulgarian Academy of Sciences.

[6] Grätzer, G., Composition of functions, Proc. Conf. on Universal Algebra, Queens Univ. Kingston Ontario 1969. 1-1o6.

[7] Lovász, L., Combinatorial problems and exercises, Akadémiai Kiadó, Budapest, 1979.

[8] Nagao, H., Multiply transitive groups, Math. Dept., California Inst. of Technology, Pasadena California 1967.

КЛОНЫ СТРОГО ТРАНЗИТИВНЫХ ГРУПП АВТОМОРФИЗМОВ

Я. Деметрович, Л. Роняи

Резюме

В статье определены свободные спектра клонов самодвойственных к строго транзитивным группам постановок. Результат обобщает прежние результаты авторов касающиеся регулярных,альтернирующих и симметрических групп. Вычисления основаны на продолжающих свойствах частных самодвойственных операций.

MTA SZTAKI Tanulmányok 147/1983 23-32

# О МОЩНОСТЯХ МНОЖЕСТВ ЗАМКНУТЫХ КЛАССОВ СОДЕРЖАЩИХСЯ В ПРЕДПОЛНЫХ КЛАССАХ В $P_k$

Я. Деметрович, Л. Ханнак

Исследовательский институт вычислительной техники
и автоматизации Венгерской Академии наук

В заметке рассматриваются вопросы связанные с изменением мощности множеств замкнутых классов содержащихся в некоторых замкнутых класса в $P_k$. Как известно, Э. Постом [10] описаны все замкнутые классы в $P_2$. Их оказалось счетное множество. В работе Ю.И. Янова и А.А. Мучника [14] установлено, что при $k \geq 3$ множество всех замкнутых классов в $P_k$ имеет мощность континуума. Окончательные результаты получились в работе [11] о числе предполных классов в $P_k$. В нашей работе изучаем мощность замкнутых классов в предполных классах, и даем окончательный ответ на этот вопрос.

Обозначим через $E_k$ множество $\{0,1,\ldots,k-1\}$, $k \geq 2$, и через $P_k$ - множество функций $f(x_1,\ldots,x_n)$, аргументы которых и сами функции принимают значения из $E_k$. Множество $P_k$ с введенными в него операциями суперпозиции называется $k$-значной логикой.

Введем следующие обозначения для семейств предполных классов, описанных в [5, 11] :

M - семейство классов монотонных функций;
U - функций, сохраняющих множества $E_k$;

S - самодвойственных функций;
L - квазилинейных функций;
C - функций, сохраняющих центральные предикаты;
B - семейство классов функций, являющихся гомоморфными прообразами классов функций, сохраняющих элементарные предикаты.

Все понятия, которые не определяются здесь, можно найти в [6, 13].

Пусть A некоторый замкнутый класс в $P_k$. Будем обозначать через $\mu(A)$ мощность множества всех замкнутых классов содержащихся в A.

Теорема 1. Если A некоторый предполный класс одного из семейств M, C, B и D, и $k \geq 3$, то $\mu(A) = \aleph_0$.

Доказательсво основывается на том, что в рассматриваемом случае в классе A может быть выделено счетное множество независимых относительно операции суперпозиции функции, аналогичное рассмотренному в работе Ю.И. Янова и А.А. Мучника [14].

Из работ С.В. Яблонского [13] и И. Розенберга [6] следует, что классы линейных функций являются предполными классами только в случае, когда k простое или степень простого числа.

Теорема 2. ([1,2]) Пусть k простое число и пусть A произвольный предполный класс всех (квази)линейных функций в $P_k$, относительно соответствующих операций. Тогда $\mu(A)$ - конечно.

Теорема 3. ([7,9])Пусть k является степенью некоторого простого числа и пусть A произвольный предполный класс всех квазилинейных функций в $P_k$ относительно соответствующих операций. Тогда $\mu(A) = \aleph_0$.

Эти утверждения следуют из более общих утверждений относящихся к классам квазилинейных функций вообще. Линейные операции в $P_k$ определяются, вообще говоря, не однозначно. Пусть A - класс всех квазилинейных функций в $P_k$, определенных некоторой линейной операцией. В работак [3, 8] показано, что если k произведение попарно различных простых чисел, то для класса A $\mu(A)$ конечно.

Для случая, когда k делится на квадрат простого числа А. Саломаа [7] показал, что $\mu(A) \geq \aleph_0$. В работе [9] Д. Лау показал, что $\mu(A) \leq \aleph_0$.

Ситуация для предполных классов самодвойственных функций в $P_k$ описывается следующими теоремами.

Теорема 4. Пусть A предполный класс самодвойственных функций в $P_3$. Тогда $\mu(A) \geq \aleph_0$.

Как известно из работ С.В. Яблонского [13] в $P_3$ имеется единственный предполный класс самодвойственных функций - класс функций самодвойственных относительно подстановки $x+1 \pmod 3$. Доказательство теоремы 4 опирается на описание всех замкнутых классов в A. Структура этого множества в значительной степени аналогична структуре множества замкнутых классов в $P_2$, хотя и имеет некоторое специфическое отличие.

Теорема 5. Пусть $k \geq 4$ и A произвольный предполный класс самодвойственных функций в $P_k$. Тогда $\mu(A) = 2$.

Доказательство теоремы опирается на следующие леммы.

Пусть $s(x)$ произвольная подстановка из $P_k$. Обозначим через $A_s$ множество всех функций самодвойственных относительно $s(x)$.

Лемма 1. Пусть подстановка $s(x)$ имеет цикл длинны не менее 5. Тогда $\mu(A_s) = 2$.

Доказательство. Пусть $s(x) = (0\,1\,2\,3\,4\,\ldots)$. Определим последовательность функций $g_i(x_1,\ldots,x_i)$ $(i \geq 3)$ самодвойственных относительно $s(x)$, следующим образом.

1. Сначала определим значения функции $g_i(x_1,\ldots x_i)$ на некотором множестве наборов $\Sigma$, состоящих из 0, 1, 2 и не переводящихся друг в друга посредством подстановки $s(x)$ и ее степеней.

а/ $g_i(0,0,\ldots,0) = 0$;

б/ если набор $(\sigma_1,\ldots,\sigma_i)\in\Sigma$ состоит из элементов 0 и 1 и содержит каждый из них, то $g_i(\sigma_1,\ldots,\sigma_i) = 0$;

в/ если набор $(\sigma_1,\ldots,\sigma_i)\in\Sigma$ состоит из элементов 0 и 2 и содержит каждый из них, то $g_i(\sigma_1,\ldots,\sigma_i) = 0$ ;

г/ обозначим через $\tilde{\sigma}^i_{m,n}\in\Sigma$ набор /длины i/ имеющий 0 в m-ом разряде, 2 в n-ом разряде и 1 в остальных i-2 разрядах.

Такие наборы будем называть характеристическими. На любом характеристическом наборе положим

$$g_i(\tilde{\sigma}^i_{m,n}) = 1;$$

д/ если набор $(\sigma_1,\ldots,\sigma_i)\in\Sigma$ состоит из элементов 0,1 и 2, содержит каждой из них и не является характеристическим, то $g_i(\sigma_1,\ldots,\sigma_i) = 0$.

2. На наборах, получающихся из наборов из $\Sigma$ с помощью перестановки $s(x)$ и ее степеней. Функция $g_i(x_1,\ldots,x_i)$ определяется на основе ее задания на множестве $\Sigma$ так, чтобы выполнялось условие самодвойственности.

3. На остальных наборах функция $g_i$ определяется произвольно, лишь бы выполнялось условие самодвойственности.

Из определения функции $g_i$ следует, что на любом наборе, состоящем из элементов 0, 1, 2 за исключением набора (2, 2,..., 2), функция $g_i$ принимает значение 0 или 2. Покажем, что функции $g_i$ независимы друг от друга в том смысле, что каждая из них не может быть выражена в виде суперпозиции остальных функций. Формула $\mathcal{D}$ над множеством функций $\{g_3,\ldots,g_{i-1},g_{i+1},\ldots\}$ будем называть i-характеристической, если она содержит лишь переменные $x_1,\ldots,x_i$ и на всех характеристических наборах /длины i/ принимает значение 1. Поскольку любая формула для $g_i$ над множеством $\{g_3,\ldots,g_{i-1},g_{i+1},\ldots\}$ является i-характеристической, то для доказательства леммы достаточно доказать следующее утверждение:

* Множество i-характеристических формул пусто.

Допустим противное, т.е. предположим, что существует i-характеристические формулы, и пусть $\mathcal{A} = g_j(\mathcal{B}_1, \ldots, \mathcal{B}_j)$ - минимальная по глубине i-характеристическая формула.

Заметим, что если для некоторого $h (1 \leq h \leq j)$ $\mathcal{B}_h$ является формулой /т.е. непеременной/, то во-первых, функция $f_h$, реализуемая этой формулой, на любом характеристическом наборе /значений переменных $x_1, \ldots, x_i$/ принимает лишь значения 0 и 1 , и во-вторых, на некотором характеристическом наборе принимает значение 0. Последнее следует из того, что $\mathcal{A}$ - минимальная по глубине i-характеристическая формула. В таблице 1 перечислены все возможности для набора выражений $\mathcal{B}_1, \ldots, \mathcal{B}_j$, определяемые наличием или отсутствием среди них некоторых переменных $x_1, \ldots, x_i$ и указан характеристический набор $\tilde{\sigma}$, на котором формула $\mathcal{A}$ принимает значение 0, тем самым будет показано, что $\mathcal{A}$ не является i-характеристической.

| | | Характеристический набор $\tilde{\sigma}$ | Свойство набора значений $\mathcal{B}_1,\ldots,\mathcal{B}_j$ |
|---|---|---|---|
| 1 | Среди $\mathcal{B}_1,\ldots,\mathcal{B}_j$ нет переменных, т.е. все они являются формулами | Набор $\tilde{\sigma}$, на котором $\mathcal{B}_1$ принимает значение 0 | состоит только из 0 и 1 и содержит 0 |
| 2 | Среди $\mathcal{B}_1,\ldots,\mathcal{B}_j$ есть некоторая переменная $x_m$, и отсутствует некоторая переменная $x_n$ | $\tilde{\sigma}^i_{m,n}$ | |
| 3 | Среди $\mathcal{B}_1,\ldots,\mathcal{B}_j$ присутствуют все переменные; и некоторая переменная $x_m$ встречается не меньше двух раз. | $\tilde{\sigma}^i_{m,n}$ | состоит из 0, 1 и 2; и имеет по крайней мере два символа 0 |
| 4 | Среди $\mathcal{B}_1,\ldots,\mathcal{B}_j$ присутствуют все переменные, и некоторые выражения $\mathcal{B}_h$ являются формулой | Набор $\tilde{\sigma}$, на котором $\mathcal{B}_h$ принимает значение 0 | |

Таблица 1.

Перечисленными случаями исчерпываются все возможности, поскольку $j \neq i$, а при $j < i$ имеет место 2. Тем самым утверждения /ж/, а с ними лемма доказана.

Лемма 2. Пусть $s(x)$ разлагается в произведение циклов и пусть длина одного из них является делителем длины другого, которая не меньше двух. Тогда $\mu(A_s) = 2$.

Пусть подстановка $s(x)$ имеет циклы $C_1$ и $C_2$ длины которых равны $|C_1|$ и $|C_2|$ и $|C_1| \leq |C_2| \geq 2$. Без ограничения общности можем считать, что подстановка $s(x)$ имеет вид $\underbrace{(0\ldots)}_{C_1}\underbrace{(12\ldots)}_{C_2}\ldots$ . Соответствующая последовательность независимых самодвойственных относительно $s(x)$ функций $g_i(x_1,\ldots,x_i)$ $(i \geq 3)$ определяется следующим образом. Функция $g_i(x_1,\ldots,x_i)$ принимает значение 1 на наборе $(\sigma_1,\ldots,\sigma_i)$, если $\sigma_\ell = 1$ $(1 \leq \ell \leq i)$ и $\sigma_t = 2$ $(t=1, \ell-1, \ell+1,..)$ На любом другом наборе всех компоненты которого принадлежат $C_1 \cup C_2$, $g_i(x_1,\ldots,x_i)$ принимает значение из $C_1$, так чтобы выполнялось условие самодвойственности. На всех остальных наборах функция $g_i$ принимает произвольное значение, лишь бы выполнялось условие самодвойственности.

Лемма 3. Пусть $s(x) = (03)(124)$. Тогда $\mu(A_s) = 2$.

Лемма 4. Пусть $s(x) = (034)(1256)$. Тогда $\mu(A_s) = 2$.

Доказательство этих лемм проводится аналогично доказательству леммы 1. Функции $g_i(x_1,\ldots,x_i)$ на наборе из элементов 0; 1 и 2 определяются также как в доказательстве леммы 1.

В заключении отметим, что утверждение теоремы 5 для $k=13, 14$ и $k \geq 16$ было установлено также С.С. Марченковым [12].

# ЛИТЕРАТУРА

[1] BAGYINSZKI J., DEMETROVICS J.,
"The structure of linear classes in prime valued logics." (Hungarian) *MTA SZTAKI Közlemények*, 16/1976, 25-52.

[2] BAGYINSZKI J., DEMETROVICS J.,
"The lattice of linear classes in prime-valued logics." *Banach Center Publications*, 8, WARSAW, PWN, 1979.

[3] BAGYINSZKI J.,
"The lattice of closed classes of linear functions over a finite ring of square-free order". *K. Marx Univ. of Economics, Dept. of Math.*, Budapest, 1979, NoDM 79-2, pp 1-21.

[4] POST, E.,
"The two-valued iterative systems of mathematical logic". *Annals of Math. Studies 5* (1941).

[5] ROSENBERG, I.G.,
"La structure des functions de plusieurs veriables sur un ensemble fini." *C.R.Acad.Sci.Paris Ser A.* 260(1965) 3817-19.

[6] ROSENBERG, I.G.,
"Über die funktionale Vollständingkeit in dem mehrwertigen Logiken." *Rozpravy Ceskoslovenské Akad. Ved. Rada Mat.Prirod* 80 (1970), 1-93.

[7] SALOMAA, A.,
"On infinitely generated sets of operations in finite algebras." *Ann.Univ. Turkuensis*, ser. A.I. 74. (1964) 1-13.

[8] A. SZENDREI
"On closed sets of linear operations over a finite set of square-free cardinality." *EIK* 14(1978) 11,547-559

[9] D. LAU.
"Über die Anzahl von abgeschlossenen Mengen von linearen Funktionen der n-wertigen logik." *EIK* 14/1978, 567-569.

[10] E. POST.
"Introduction to a general theory of elementary propositions." *Amer. J. Math.* 93(1921) 183-185.

[11] Захаров Е.Ю., Кудрявцев В.Б., Яблонский С.В.
"О предполных классах в k-значных логиках".
Докл. АН СССР 186(1969) 3, 509-512.

[12] Марченков С.С.,
"О самодвойственных замкнутых классах в k-значных логиках". Пробл. Киб. 36(1979).

[13] Яблонский С.В.
"Функциональные построения в k-значной логике".
Труды Мат. инст. имю Стеклова 51(1958) 5-142.

[14] Янов Ю.И., Мучник А.А.
"О существовании k-значных классов, не имеющих конечного базиса". Докл. АН СССР 127(1959) 1, 144-146.

## ON CARDINALITY OF THE SETS OF CLOSED CLASSES BEING IN PRE-COMPLETE CLASSES IN $P_k$

J. Demetrovics - L. Hannák

Abstract

In the paper the question of the cardinality of closed classes that are contained in some closed classes in $P_k$, is investigated. Final answers to some open questions are given.

MTA SZTAKI Tanulmányok 147/1983 33-45

# МАКСИМАЛЬНО ВЫДЕЛИМЫЕ МНОЖЕСТВА АРГУМЕНТОВ ФУНКЦИЙ

К.Н. Чимев

Филиал Софийского Университета, Благоевград

Рассматриваются вопросы о максимально выделимых множествах аргументов функций, по отношению одних либо других множеств аргументов функций.

Используется терминология из [1 - 19].

Если $f$ функция, то множество ее существенных аргументов будем обозначать $R_f$ .

Если $f$ функция и

$$R_1 \subset R_f,\ R_2 \subset R_f,\ R_1 \cap R_2 = \phi,\ R_1 \neq \phi,\ R_2 \neq \phi,$$

то множество $R_1$ называется выделимым для $f$ по отношению $R_2$, если для переменных от $R_2$ существуют такие значения, что после замещения ими от $f$ получается такая функция $f^*$ , что

$$R_{f^*} \supset R_1.$$

Если $f$ функция и $R_1 \subset R_f (R_1 \neq \phi)$ , то будем считать, что $R_1$ выделимо для $f$ по отношению пустого множества $\phi$ .

Если $f$ функция и $R_1 \subset R_f (R_1 \neq \phi)$ , то $R_1$ называется выделимым для $f$ , если $R_1$ выделимо для $f$ по отношению $R_f \setminus R_1$.

Множество всех выделимых для функции $f$ множеств существенных переменных обозначим $S_f$ .

Если $f$ функция и

$$R \subset R_f,\ R_2 \subset R_f,\ R \neq \phi,\ R \cap R_2 = \phi,$$

то множество $R_1 (R_1 \subset R,\ R_1 \neq \phi)$ называется максимально вы-

делимым для $f$ подмножеством множества $R$ по отношению $R_2$, если $R_1$ выделимо для $f$ по отношению $R_2$ и не существует множество $R^*$, которое выделимо для $f$ по отношению $R_2$ и

$$R_1 \subset R^* \subset R, \quad R_1 \neq R^*.$$

Если $f$ функция и

$$R \subset R_f, \; R_2 \subset R_f, \; R \neq \emptyset, \; R \cap R_2 = \emptyset,$$

то множество $R_1$ $(R_1 \subset R, \; R_1 \neq \emptyset)$ называется максимально выделимым для $f$ подмножеством множества $R$ по отношению $(R \setminus R_1) \cup R_2$, если $R_1$ выделимо для $f$ по отношению $(R \setminus R_1) \cup R_2$ и не существует множество $R^*$, которое выделимо для $f$ по отношению $(R \setminus R^*) \cup R_2$ и

$$R_1 \subset R^* \subset R, \quad R_1 \neq R^*.$$

Пусть $f$ функция $(|R_f| \geq 3)$ и

$$R \subset R_f, \; R \neq \emptyset, \; R \neq R_f.$$

Множество $R_1$ называется максимально выделимым для $f$ подмножествам множества $R$, если $R_1$ максимально выделимо для $f$ подмножество множества $R$ по отношению множества

$$(R \setminus R_1) \cup (R_f \setminus R) = R_f \setminus R_1.$$

Теорема 1. Если $f$ функция и

$$R \subset R_f, \; R_2 \subset R_f, \; R \neq \emptyset, \; R_2 \neq \emptyset, \; R \cap R_2 = \emptyset,$$

то множество $R_1$ максимально выделимо для $f$ подмножество множества $R$ по отношению $(R \setminus R_1) \cup R_2$, тогда и только тогда, когда $R_1$ максимально выделимо для $f$ подмножество множества $R$ по отношению $R_2$.

Следствие 1. Если $f$ функция $(|R_f| \geq 3)$ и $R \subset R_f (R \neq \phi, \ R \neq R_f)$, то множество $R_1 (R_1 \subset R)$ максимально выделимо для $f$ подмножество множества $R$, тогда и только тогда, когда $R_1$ максимально выделимо для $f$ подмножество множества $R$ по отношению множества $R_f \setminus R$.

Теорема 2. Пусть $f$ функция $(|R_f| \geq 3)$ и

$$R \subset R_f, \quad R \neq \phi, \quad R \neq R_f.$$

Если $R_1$ максимально выделимо для $f$ подмножество множества $R$, по отношению некоторого множества $R_2 (R_2 \subset R_f \setminus R, \ R_2 \neq \phi)$, то для каждого выбора переменных от $R \setminus R_1$ функция $f^*$, которая получается от $f$ после замены этих переменных с произвольными значениями зависит существенно, хотя бы от одной переменной от $R_2$ и множество $R_1$ выделимо для нее по отношению множества $R_{f^*} \cap R_2$.

Доказательство. При условии теоремы, рассмотрим нетривиальный случай, когда множество $R$ не выделимо для $f$ по отношению $R_2$.

Пусть $R_1$ максимально выделимо для $f$ подмножество множества $R$ по отношению некоторого множества $R_2 (R_2 \subset R_f \setminus R, \ R_2 \neq \phi)$. При этом $R_1 \neq R$.

Не ограничивая общности рассмотрения, пусть например

$$R_1 = \{x_1, \dots, x_m\},$$

$$R = \{x_1, \dots, x_m, x_{m+1}, \dots, x_{m+p}\}, \ 2 \leq m+p \leq n-1, \ p \geq 1,$$

$$R_2 = \{x_{m+p+1}, \dots, x_{m+p+r}\}, \quad m+p+r \leq n.$$

Допустим, что существуют переменные от $R \setminus R_1$, например $x_{m+1}, \dots, x_{m+t}$, $1 \leq t \leq p$, и существуют такие значения

для них $c'_{m+1}, \dots, c'_{m+t}$ , что

$$R_{f_1} \cap R_2 = \phi,$$

где

$$f_1 = f(x_{m+1} = c'_{m+1}, \dots, x_{m+t} = c'_{m+t}).$$

По условию множество $R_1$ максимально выделимо для $f$ подмножество множества $R$ по отношению $R_2$ . Тогда существуют такие значения $c'_{m+p+1}, \dots, c'_{m+p+r}$ для $x_{m+p+1}, \dots, x_{m+p+r}$, что

$$R_{f_2} \supset R_1 \quad \text{и} \quad R_{f_2} \cap (R \setminus R_1) = \phi,$$

где

$$f_2 = f(x_{m+p+1} = c'_{m+p+1}, \dots, x_{m+p+r} = c'_{m+p+r}).$$

В таком случае

$$R_{f_3} \supset R_1,$$

где

$$f_3 = f_2(x_{m+1} = c'_{m+1}, \dots, x_{m+t} = c'_{m+t})$$

$$= f_1(x_{m+p+1} = c'_{m+p+1}, \dots, x_{m+p+r} = c'_{m+p+r}).$$

Следовательно

$$R_{f_1} \supset R_1.$$

В соответствии с сделонным предположением

$$R_{f_1} \cap \{x_{m+p+1}, \dots, x_{m+p+r}\} = \phi .$$

Поэтому для всех значений $c_{m+p+1}, \dots, c_{m+p+r}$ для $x_{m+p+1}, \dots, x_{m+p+r}$ функция $f(x_{m+p+1}=c_{m+p+1}, \dots, x_{m+p+r}=c_{m+p+r})$ будет зависеть существенно от всех переменных от $R_1$ .

Пусть $c''_{m+p+1}, \dots, c''_{m+p+r}$ такие значения для $x_{m+p+1}, \dots, x_{m+p+r}$, что $x_{m+1} \in R_{f_4}$, где

$$f_4 = f(x_{m+p+1}=c''_{m+p+1}, \dots, x_{m+p+r}=c''_{m+p+r}).$$

Множество

$$R_1^* = R \cap R_{f_4}$$

выделимо для $f$ по отношению $R_2$ . При этом

$$R \supset R_1^* \supset R_1 \cup \{x_{m+1}\},$$

что противоречит первоначальному условию. Следовательно сделанное предположение неверно.

Докажем вторую часть заключения теоремы. Рассмотрим какие-нибудь переменных от $R \setminus R_1$ , например $x_{m+1}, \dots, x_{m+t}$, $1 \leqq t \leqq p$ . Пусть $c''_{m+1}, \dots, c''_{m+t}$ произвольные значения для $x_{m+1}, \dots, x_{m+t}$. Надо доказать, что если

$$f^* = f(x_{m+1}=c''_{m+1}, \dots, x_{m+t}=c''_{m+t}),$$

то множество $R_1$ выделимо для $f^*$ по отношению $R_{f^*} \cap R_2$ .

Пусть $c^*_{m+p+1}, \dots, c^*_{m+p+r}$ такие значения для

$x_{m+p+1}, \dots, x_{m+p+r}$, что

$$R_{f_5} \supset R_1,$$

где

$$f_5 = f\left(x_{m+p+1} = c^*_{m+p+1}, \dots, x_{m+p+r} = c^*_{m+p+r}\right).$$

Тогда

$$R_{f_5} \cap \{x_{m+1}, \dots, x_{m+p}\} = \phi.$$

Поэтому, если

$$f_6 = f_5\left(x_{m+1} = c''_{m+1}, \dots, x_{m+t} = c''_{m+t}\right)$$

$$= f^*\left(x_{m+p+1} = c^*_{m+p+1}, \dots, x_{m+p+r} = c^*_{m+p+r}\right),$$

то

$$R_{f_6} \supset R_1$$

и

$$R_{f_6} \cap \left(R \setminus \{x_1, \dots, x_{m+t}\}\right) = \phi.$$

Уже доказали, что

$$R_2' = R_2 \cap R_{f^*} \neq \phi.$$

Пусть, например

$$R_2' = \{x_{m+p+1}, \dots, x_{m+p+\ell}\}, \quad 1 \leqq \ell \leqq r.$$

Функция

$$f^*\left(x_{m+p+1}=c^*_{m+p+1},\dots,x_{m+p+\ell}=c^*_{m+p+\ell}\right)=$$

$$=f^*\left(x_{m+p+1}=c^*_{m+p+1},\dots,x_{m+p+r}=c^*_{m+p+r}\right)$$

зависит существенно от каждой переменной от $R_1$. Поэтому множество $R_1$ выделимо для $f^*$ по отношению множества $R'_2$.

Теорема доказана.

Можно поставить вопрос: всегда ли, когда $R_1$ максимально выделимо для $f$ подмножество множества $R\,(R\subset R_f, R\neq\phi, R\neq R_f)$ по отношению некоторого множества $R_2\,(R_2\subset R_f\setminus R, |R_2|\geq 2)$, для каждого выбора переменных от $R\setminus R_1$, каждая функция $f^*$, которая получена от $f$ после замены этих переменных с произвольными допустимыми для них значениями, зависит существенно не менее чем от двух переменных?

С помощью примеров можно показать, что ответ на этот вопрос отрицательный.

Следствие 1. Пусть $f$ функция $(|R_f|\geq 3)$ и $R\subset R_f$, $R\neq\phi$, $R\neq R_f$.

Если $R_1$ максимально выделимо для $f$ подмножество множества $R$, то для каждого выбора переменных от $R\setminus R_1$ функция $f^*$, которая получается от $f$ после замены этих переменных с произвольными значениями, зависит существенно не менее чем от одной переменной от $R_f\setminus R$ и $R_1\in S_{f^*}$.

Следствие 2. Если $f$ функция $(|R|\geq 3)$ и множество $R_2\,(R_2\neq\phi,\ R_2\subset R_f)$ не выделимо для $f$ по отношению

множества $R\,(R \neq \phi,\ R \subset R_f,\ R \cap R_2 = \phi)$ и множество $R_1 (R_1 \subset R_2)$ выделимо для $f$ по отношению $R$, то существует такая переменная $x_i \in R_2 \setminus R_1$, что для каждого значения $c_i$ для $x_i$ функция $f(x_i = c_i)$ зависит существенно не менее чем от одной переменной от $R$ и $R_1$ выделимо для неё по отношению $R_{f(x_i = c_i)} \cap R$.

Следствие 3. Если $f$ функция $(|R_f| \geq 3)$ и

$$R_1 \in S_f,\ R_2 \notin S_f\ (R_1 \subset R_2 \subset R_f,\ R_1 \neq \phi),$$

то существует такая переменная $x_i \in R_2 \setminus R_1$, что для каждого значения $c_i$ для $x_i$,

$$R^* = R_{f(x_i = c_i)} \cap (R_f \setminus R_2) \neq \phi$$

и $R_1$ выделимо для $f(x_i = c_i)$ по отношению $R^*$.

Теорема 3. Пусть $f$ функция $(|R_f| \geq 3)$ и

$$R \subset R_f,\ R_2 \subset R_f,\ R \neq \phi,\ R_2 \neq \phi,\ R \cap R_2 = \phi.$$

Если $R_1$ максимально выделимо для $f$ подмножество множества $R$ по отношению $(R \setminus R_1) \cup R_2$, то для каждого выбора переменных от $R \setminus R_1$ функция $f^*$, которая получается от $f$ после замены этих переменных с произвольными значениями зависит существенно, хотя бы от одной переменной от $R_2$ и множество $R_1$ выделимо для нее по отношению множества её существенных переменных, которые принадлежат множеству $(R \setminus R_1) \cup R_2$.

Теорему 3 можно доказать с помощью теорема 2.

Следствие 1. Пусть $f$ функция $(|R_f| \geq 3)$ и

$$R \subset R_f,\ R_2 \subset R_f,\ R \neq \phi,\ R_2 \neq \phi,\ R \cap R_2 = \phi.$$

Если $R$ не выделимо для $f$ по отношению $R_2$ и $R_1 (R_1 \subset R, R_1 \neq \phi)$ выделимо для $f$ по отношению $(R \setminus R_1) \cup R_2$, то существует такая переменная $x_i \in R \setminus R_1$, что для каждого значения $c_i$ для $x_i$ функция $f(x_i = c_i)$ зависит существенно не менее чем от одной переменной от $R_2$ и $R_1$ выделимо для неё по отношению множества ее существенных переменных которые принадлежат $(R \setminus R_1) \cup R_2$.

Следствие 2. Если $f$ функция $(|R_f| \geq 3)$ и

$$R_1 \in S_f,\ R_2 \notin S_f\ (R_1 \subset R_2 \subset R_f),$$

то существует такая переменная $x_i \in R_2 \setminus R_1$, что для каждого значения $c_i$ для $x_i$ функция $f(x_i = c_i)$ зависит существенно не менее чем от одной переменной от $R_f \setminus R_2$ и $R_1 \in S_{f(x_i = c_i)}$.

Следствие 3. Если $f$ функция $(|R_f| \geq 3)$ и $R \notin S_f\ (R \subset R_f, |R| \geq 2)$, то:

а) Для каждой переменной $x_i \in R$ существует множество $R_1 \in S_f (R_1 \subset R)$ которое содержит $x_i$ и для каждого такого множества существует переменная $x_j \in R \setminus R_1$, что для каждого значения $c_j$ для $x_j$ функция $f(x_j = c_j)$ зависит существенно не менее чем от одной из переменных от $R_f \setminus R$ и $R_1 \in S_{f(x_j = c_j)}$.

б) Существуют не менее чем две подмножества $R_1$ и $R_2$ множества $R$, такие что для каждого $i = 1, 2$ существует переменная $x_j \in R \setminus R_i$, что для каждого значения $c_j$ для $x_j$ функция $f(x_j = c_j)$ зависит существенно не менее чем от одной

из переменных от $R_f \setminus R$ и $R_i \in S_{f(x_j = c_j)}$

в) Для каждой переменной $x_i \in R$ существует такая переменная $x_j \in R$, что для каждого значения $c_j$ для $x_j$, $x_i \in R_{f(x_j = c_j)}$ и $R_{f(x_j = c_j)} \cap (R_f \setminus R) \neq \emptyset$.

Следствие 4. Пусть $f$ функция $(|R_f| \geq 3)$ и $R \in S_f$, $1 \leq |R| \leq n-2$.

Если существует такая переменная $x_i \in R_f \setminus R$, что $R \cup \{x_i\} \notin S_f$, то для каждого значения $c_i$ для $x_i$ функция $f(x_i = c_i)$ зависит существенно не менее чем от одной переменной от $R_f \setminus (R \cup \{x_i\})$ и $R \in S_{f(x_i = c_i)}$. Если $|R| = n-2$, то переменная $x_i$ c-сильно существенная для $f$.

Следствие 5. Если $f$ функция $(|R_f| \geq 3)$ и $\{x_i, x_j\} \notin S_f$, $\{x_i, x_j\} \subset R_f$, то для каждого значения $c_j$ для $x_j$ функция $f(x_j = c_j)$ зависит существенно от $x_i$ и зависит существенно не менее чем от одной переменной от $\{x_1, \ldots, x_n\} \setminus \{x_i, x_j\}$.

Следствие 6. Если $f$ функция $(|R_f| \geq 4)$ и $R \notin S_f$, $|R| = 3$, то хотя бы одна из переменных от $R$ c-сильно существенная для $f$ по отношению $R$. Если $|R_f| = 4$, то хотя бы одна из переменных от $R$ c-сильно существенная для $f$.

# Л И Т Е Р А Т У Р А

1. Яблонский С. В. Функциональные построения в к-значной логике. Труды математического института им. В.А. Стеклова, т. 51, 1958, 5 - 142.
2. Яблонский С.В., Г.П. Гаврилов, В.Б. Кудрявцев. Функции алгебры логики и классы Поста, Москва, 1966.
3. Schwartz R.E. Existence and uniquense properties of boolean functions. SIAM Jurnal on applied Matematics, 1970, 18, 2, 454 - 461.
4. Solomaa A. On essential variables of functions, especially in the algebra of logic. Annales academiae scientiarum fennicae, ser. A, 339 (1963), 1 - 11.
5. Лупанов О. Б. Об одном классе схем из функциональных элементов. Сб. "Проблемы кибернетики", вып. 7, 1962,61-114.
6. Соловьев Н.А. К вопросу о существенной зависимости функции алгебры логики. Сб. "Проблемы кибернетики", вып. 9, 1963, 333 - 335.
7. Брейтбарт Ю.Я. О существенных переменных функции алгебры логики, ДАН СССР, 1967, т. 172, № 1, 9-10.
8. Чимев К. Н. Върху зависимостта на функциите от $P_k$ от аргументите им. Год. на ВТУЗ. Математика, т.IV, кн.3, 1967, 5-13.
9. Чимев К. Н. Върху зависимостта на функциите от к-значната логика от аргументите им. Год. на ВТУЗ. Математика, т.VI, кн. 2, 1970, 53 - 62.
10. Чимев К. Н. Върху някои свойства на функциите. Год. на ВТУЗ. Математика, т.VII, кн. 1, 1971, 23 - 32.
11. Чимев К. Н. Върху инвариантността на отделимите двойки на функциите. Год. на ВТУЗ. Математика, т.VIII, кн. 1, 1972, 123 - 136.

12. Чимев К. Н. Върху подфункциите и силно съществените променливи на функциите. Год. на ВТУЗ. Математика, т. IX, кн. 4, 1973, 43 - 55.
13. Чимев К. Н. Върху отделимите двойки на функциите. Год. на ВТУЗ, Математика, т.VII, кн. 3, 1971, 7 - 12.
14. Чимев К. Н. Върху отделимите подмножества и силно съществените променливи на функциите. Год. на ВУЗ. Приложна математика, т. X, кн. 4, 1974, 7 - 13.
15. Čimev K.N. On some properties of functions. Colloquia Math. Soc. János Bolyai. Finite algebra and multiple-valued logic. Szeged, 1979, 97 - 110.
16. Чимев К. Н. О выделимых множествах аргументов функций. MTA Sz TAKI, Közlemények, 24, 1980, 19 - 27.
17. Чимев К. Н. Об одном классе функций. Rostock. Math. Kolloq. 19, 1982, 9- 17.
18. Чимев К. Н. О выделимых множеств аргументов некоторых функций. Abstracts of 7 conference on operating systems. Visegrad, 1982, 12 - 14.
19. Чимев К. Н. Подфункции и отделими множества от аргументи на функциите. Математика и математическо образование. С., БАН, 1982, 105 - 122.

# MAXIMAL SEPARABLE SETS OF ARGUMENTS OF THE FUNCTIONS

K. N. Cimev

(Abstract)

The article discusses the questions of the maximal separable sets of arguments of the functions with respect to some sets or other of the arguments of the functions.

If $f$ is a function , then the set of its essential variables is marked with $R_f$ .

Let $f$ be a function $(|R_f| \geq 3)$ and

$$R \subset R_f,\ R_2 \subset R_f,\ R \neq \emptyset,\ R_2 \neq \emptyset,\ R \cap R_2 = \emptyset.$$

Then the followig two theorems are true.

Theorem 1. The set $R_1$ is maxmal separable for $f$ subset of $R$ with respect to $(R \setminus R_1) \cup R_2$ then and only then, when $R_1$ is maximal separable for $f$ subset of $R$ with respect to $R_2$ .

Theorem 2. If the set $R_1$ is maximal separable for $f$ subset of $R$ with respect to the set $R_2$ , then for every choice of variables from $R \setminus R_1$ the function $f^*$ which is obtained from $f$ after replacing these variables with their arbitrary values, depends essentially at least one of the variables from $R_2$ and the set $R_1$ is separable for $f^*$ with respect to $R_{f^*} \cap R_2$ .

The truth of a series of other statements follows from theorems 1 and 2.

МТА SZTAKI Tanulmányok 147/1983 47-50

# О ВЫДЕЛИМЫХ ПОДМНОЖЕСТВАХ АРГУМЕНТОВ ФУНКЦИЙ ИЗ $P_k$

Й.Д. Денев - И.Д. Гюдженов

Институт математики с ВЦ Болгарской Академии наук,
Филиал Софийского университета, Благоевград.

Впервые понятие выделимой пары функции алгебры логики введено Ю. Я. Брейбартом [1]. Несколько позднее в качестве обобщения К. Н. Чимев [2] , [3] вводит понятие выделимой $m$-торки для произвольных функций, показывая при этом, что существуют функции, имеющие невыделимые $m$-торки. Возникает вопрос - каково число функций, у которых любое подмножество их аргументов является выделимым.

В настоящей работе показано, что для "почти всех" функций $k$-значной логики выделимо любое подмножество их аргументов.

Определение. Функция $f(x_1,x_2,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_n)$ из $P_k$ называется существенно зависящей от аргумента $x_i$ , если найдутся такие наборы $\tilde{\alpha}$ и $\tilde{\beta}$ , что

$$\tilde{\alpha} = (\alpha_1,\alpha_2,\ \ldots,\ \alpha_{i-1},\alpha_i,\alpha_{i+1},\ldots,\alpha_n),$$
$$\tilde{\beta} = (\alpha_1,\alpha_2,\ \ldots,\ \alpha_{i-1},\beta_i,\alpha_{i+1},\ldots,\alpha_n),\ \alpha_i \neq \beta_i,$$

и $f(\tilde{\alpha}) \neq f(\tilde{\beta})$ , $\alpha_j \in \{0,1,\ldots,k-1\}, \beta_i \in \{0,1,\ldots,k-1\}$.

Определение. Множество $\{x_{i_1}, x_{i_2},\ldots,x_{i_m}\}$ называется выделимым для функции $f(x_1,x_2,\ldots,x_n)$, если существуют $n-m$ констант, таких что, при замене ими переменных из множества $\{x_1,x_2,\ldots,x_n\} \setminus \{x_{i_1},x_{i_2},\ldots,x_{i_m}\}$ функция $f$ существенно зависит от $x_{i_1},x_{i_2},x_{i_3},\ldots,x_{i_m}$.

Пусть $\{x_{i_1},x_{i_2},\ldots,x_{i_m}\}$ выделимое для $f(x_1,x_2,\ldots,x_n)$ , $n \geq 2$. Без ограничения общности примем для простоты, что таким является множество $\{x_{n-m+1},\ldots,x_n\}$.

Используя хорошо известный аналог разложения Шенона, представим функцию $f$ в виде:

$$f = \max_{(\sigma_1,\ldots,\sigma_{n-1})} \{\min[I_{\sigma_1}(x_1),\ldots, I_{\sigma_{n-m}}(x_{n-m}), f(\sigma_1,\sigma_2,\ldots,\sigma_{n-m},x_{n-m+1},\ldots,x_n)]\},$$

где

$$I_i(x)=\begin{cases} k-1 & \text{для } x = i, \\ 0 & \text{для } x \neq i. \end{cases}$$

Ясно, что $\{x_{n-m+1},\ldots,x_n\}$ не является выделимым для $f$ тогда и только тогда, когда все функции

$$f(\sigma_1,\sigma_2,\ldots,\sigma_{n-m}, x_{n-m+1},\ldots, x_n)$$

не зависят существено от всех своих переменных. В противном случае множество $\{x_{n-m+1},\ldots,x_n\}$ выделимо для $f$. Зафиксируем набор $(\sigma_1,\sigma_2,\ldots,\sigma_{n-m})$. Число функций вида $f(\sigma_1,\sigma_2,\ldots,\sigma_{n-m},x_{n-m+1},\ldots,x_n)$ зависящих не более чем $m-1$ переменных, не больше чем $m\,k^{k^{m-1}}$. Число всех функций из $P_k$, для которых множество $\{x_{n-m+1},\ldots,x_n\}$ не является выделимым, не больше числа возможных $k^{n-m}$-торок, получаемых из $m\,k^{k^{m-1}}$ функций, т. е., не больше чем

$$(m\,k^{k^{m-1}})^{k^{n-m}} = (k^{k^{m-1}+\log_k m})^{k^{n-m}} = k^{k^{n-1}+k^{n-m}\log_k m}$$

Означим через $P_{k,m}(x^{(n)})$ множество всех функций из $P_k$, , для которых не выделимо хотя бы одно множество из $m$ переменных, а через $P_k(x^{(n)})$ - множество всех функций, для которых не выделимо хотя бы одно подмножество их аргументов.

Теорема 1. У почти всех функций из $P_k$ выделимы все множества из $m$ переменных.

Доказательство. Очевидно,

$$|P_{k,m}(x^{(n)})| \leq \binom{n}{m}(k^{k^{n-1}+k^{n-m}\log_k m}) < 2^n(k^{k^{n-1}+k^{n-m}\log_k m}),$$

Следовательно,

$$\lim_{n\to\infty} \frac{|P_{k,m}(x^{(n)})|}{k^{k^n}} \to 0,$$

ч.т.д.

Теорема 2. У почти всех функций из $P_k$ все множества переменных.

Доказательство. Имеем

$$|P_k(x^{(n)})| \leq \sum_{m=2}^{n-1} 2^n (k^{k^{n-1}} + k^{n-m} \log_k m) < k^{k^n} \left(\frac{\log_k 2+k}{k^2}\right) + n \log_k 2 + \log_k n .$$

Путем несложных вычислений можно показать, что

$$\lim_{n\to\infty} \frac{|P_k(x^{(n)})}{k^{k^n}} \to 0,$$

ч.т.д.

## Л И Т Е Р А Т У Р А


1. Брейтбарт, Ю. Я., О существенных переменных функциях алгебры логики, Доклады АН СССР, 1967, т. 172 № 1, 9 - 10.
2. Чимев К. П., Върху силно съществените променливи на функциите от $P_k$, Год. на ВТУЗ, Математика т. V, кн. 2, 1968.
3. Чимев К. П., Върху отделимите подмножества и силно съществените променливи на функциите, Год. на ВТУЗ, Приложна математика, т. X кн. 4, 1974 .

# ON THE SEPARABLE SUBSETS OF VARIABLES FOR FUNCTIONS FROM $P_k$

J.D. Denev, I.D.Gjudjenov

Abstract

The subset S is called separable for $f(x_1,x_2,\ldots,x_n) \in P_k$ iff after some replacing variables from S by constants a subfunction of f is obtained which depends on all the remaining variables.

The main result if this paper is - almost all functions from $P_k$ have separable subsets of variables only.

MTA SZTAKI Tanulmányok 147/1983

51-59

# О ВЫДЕЛИМЫХ ПАРАХ ОДНОГО КЛАССА ФУНКЦИЙ

И. Д. Гюдженов

Филиал Софийского университета, Благоевград

В работе рассматривается вопрос о выделимых парах функций алгебры логики, существенно зависящих от семи переменных, а также имеющих переменные первого или второго порядка.

Случай, когда функции имеют менее семи существенно зависящих переменных рассмотрен К.Чимевым [3,7].

Определение 1. Пара переменных $(x_i, x_j)(1 \leqq i < j \leqq n)$, $f(x_1, x_2, \ldots, x_n)$ $(n > 1)$ называется выделимой, если существует набор из $(n-2)$ таких констант, что при подстановке этих констант на место всех переменных $f(x_1, x_2, \ldots, x_n)$, кроме $x_i$ и $x_j$, получается функция, существенно зависящая от $x_i$ и $x_j$.

Определение 2. Переменная $x_i$ $(1 \leqq i \leqq n)$ функции $f(x_1, x_2, \ldots, x_n)(n > 1)$ назнвается переменной $k$-го порядка, если существует переменные $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ $(1 \leqq i_1 < \cdots < i_k \leqq n)$, $i_t \neq i, (t = 1, 2, \ldots, k)$ такие, что каждая пара $(x_i, x_{i_t})$ выделима и, каково бы ни было $j \neq i_t$, $j \neq i$, пара $(x_i, x_j)$ не является выделимой.

Обозначим через $P_{2,7}$ множество всех функций алгебры логики, которые существенно зависят от семи переменных.

Обозначим также, через $Sf$ множество выделимых пар функции $f$.

Под графом некоей функции подразумевают неориентированный граф с вершинами - существенными переменнами, и ребрами - выделимыми парами.

Пусть $f \in P_2$ имеет переменную первого порядка. Согласно теореме 6 [2] следует, что она единственна.

Будем считать, что эта переменная есть $x_1$ и она образует выделимую пару с $x_2$ .

Из теоремы 2 [2] следует, что $x_2$ - переменная шестого порядка. Кроме того, доказано (лемма 1 [8] и лемма 7 [9]), что

1) $f$ представима в виде

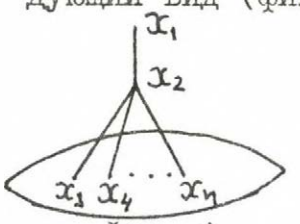
$$f(x_1,x_2,\dots,x_n)=\varphi(x_1,x_2)+\psi(x_2,x_3,\dots,x_n)$$

2) Каждая пара вида $(x_2,x_j)$, $i,j\in\{3,4,\dots,n\}$ выдилима для $f$ .

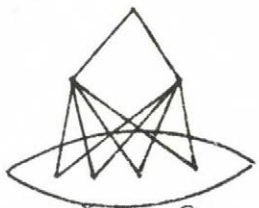
3) Пара $(x_i,x_j)$ , $i,j\in\{3,4,\dots,n\}$ выделима для тогда и только тогда, когда выделима для $\psi$ .

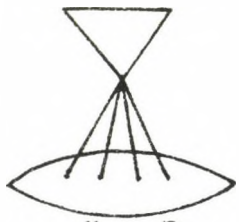
Так как $f$ зависит существенно от всех своих переменных, то для $x_2$ существует такое $c_2\in\{0;1\}$ значение, что функция $f(x_2=c_2)=\psi(x_2=c_2)$ зависит существенно от $x_3$ , $x_4$ ,..., $x_n$ . Тогда выделимыми для $f$ будут пары $(x_1,x_2)$ , $(x_2,x_3)$, $(x_2,x_4)$ ,..., $(x_2,x_n)$, а также все пары выделимые для $\psi(x_2=c_2)$ .

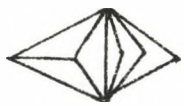
Следовательно, граф функции $f$ будет иметь следующий вид (фиг. 1).



фиг. 1


фиг. 2


фиг.3


фиг.4

Здесь, подграф функции $f$ , с вершинами элементами множества $\{x_3,x_4,\dots,x_n\}$ является графом функции $(n-2)$ переменных.

Таким образом мы доказали следующую теорему.

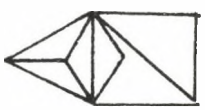
Теорема 1. Если для функции $f\in P_2$ , $x_1$ есть переменная первого порядка и $(x_1,x_2)\in Sf$, то граф $f$ имеет вышеуказанный вид (фиг.1), где подграф $f$ с вершинами

елементами множества $\{x_3, x_4, \ldots, x_n\}$ является графом функции $(n-2)$ переменных.

Следствие. Если для функция $f \in P_{2,7}$, $x_1$ есть переменная первого порядка и $(x_1, x_2) \in Sf$, то граф $f$ имеет вышеуказанный вид (фиг.1), где подграф $f$ с вершинами елементами множества $\{x_3, x_4, \ldots, x_7\}$ является графом функции пяти переменных.

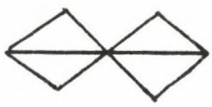
Лемма. Не существует функция $f(x_1, x_2, \ldots, x_n)$, где $f \in P_k$ и $n \geqslant k^2+3$, имеющая две переменные $(n-1)$ порядка и одновременно с этим, имеющая $(n-2)$ переменные второго порядка, необразующие между собой выделимые пары.

Теорема 2. Если у функции $f \in P_{2,7}$ не существует переменная первого порядка, но при этом имеет переменные второго порядка, то для ее графа невозможен случай отличный от тех, которые указаны на фиг. 3 - 50.

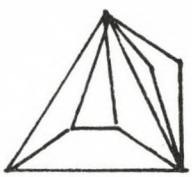


фиг.5

фиг.6



фиг.7

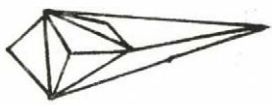


фиг.8

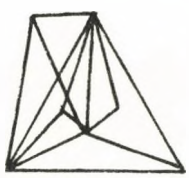
фиг.9

фиг.10

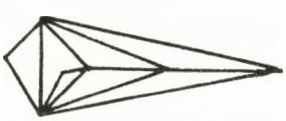
фиг.11



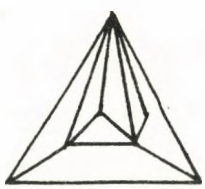
фиг.12



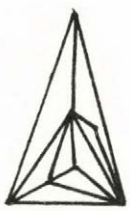
фиг.13



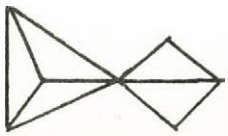
фиг.14



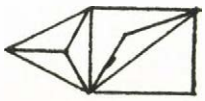
фиг.15



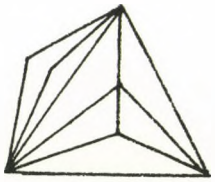
фиг.16



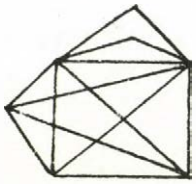
фиг.17



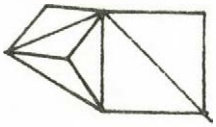
фиг.18



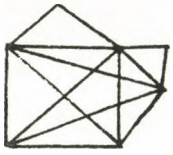
фиг.19



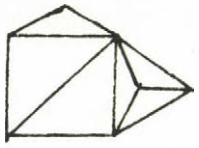
фиг.20



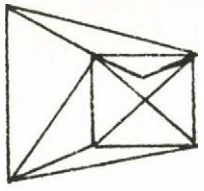
фиг.21



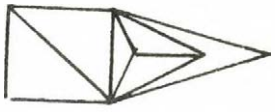
фиг.22



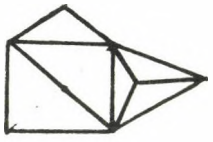
фиг.23



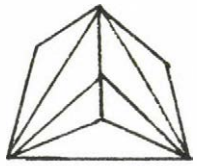
фиг.24



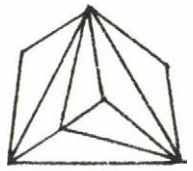
фиг.25



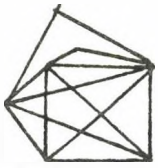
фиг.26



фиг.27



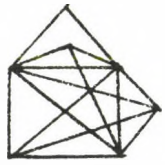
фиг.28



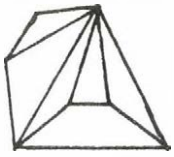
фиг.29

фиг.30

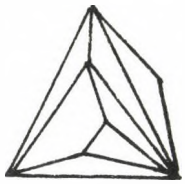
фиг.31



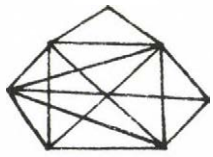
фиг.32



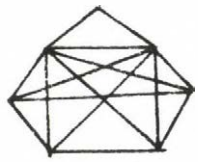
фиг.33



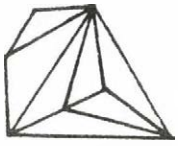
фиг.34



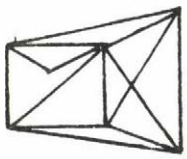
фиг.35



фиг.36

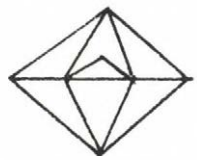


фиг.37



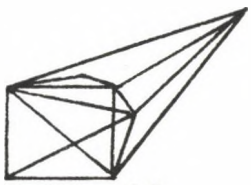
фиг.38

фиг.39

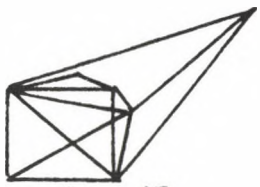


фиг.40

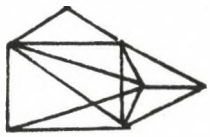
фиг.41



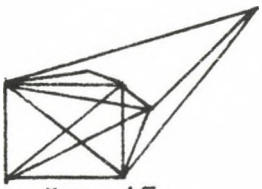
фиг.42



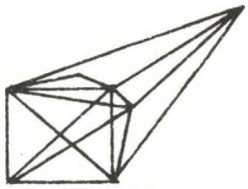
фиг.43



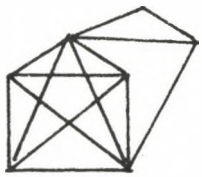
фиг.44



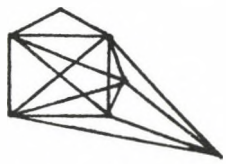
фиг.45



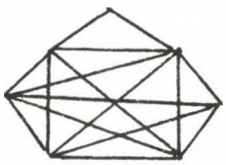
фиг.46



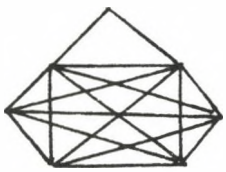
фиг.47



фиг.48



фиг.49



фиг.50

Несколько замечаний относительно доказательства теоремы 2.

Рассматривается два основных случая.

1) Случай, когда переменная второго порядка образует выделимые пары с переменными, не образующими между собой выделимую пару.

Из теоремы 1,2,3,4 [7] и теоремы 1 [6] следует, что граф $f$ имеет вышеуказанный вид (фиг.2), где подграф $f$ с вершинами елементами множества $\{x_4, x_5, x_6, x_7\}$ является графом четырех переменных.

2) Случай, когда переменная второго порядка образует выделимые пары с переменнами, образующими между собой выделимую пару.

Пусть $x_1$ есть переменная второго порядка, а также $(x_1, x_2), (x_1, x_3), (x_2, x_3) \in Sf$.

а) Если только одна из переменных $x_2$ , $x_3$ является переменной второго порядка, то граф $f$ имеет вышеуказанный вид (фиг.3), где подграф $f$ с вершинами елементами множества $\{x_4, x_5, x_6, x_7\}$ является графом четырех переменных.

б) Если каждая из переменных $x_3$ , $x_4$ имеет порядок не меньше чем три, то для графа $f$ невозможен случай отличный от тех, каторые указаны на фиг. 4 - 50.

Отметим, что автору не удалось построить функцию с графом вида фиг.40, или доказать, что такая функция не существует.

Примерами функции, каторые имеют вышеуказанные виды фиг. 1-39 и фиг. 41-50, являются следущие функции из алгебры логики:

$$f_1 = x_1\bar{x}_2 + x_2x_3x_4x_5x_6x_7 \pmod 2$$

$$f_2 = 1 + x_1 + x_2 + x_1x_2 + x_1x_3 + (x_2 + x_3)x_4x_5x_6x_7 \pmod 2$$

$$f_3 = x_1x_2x_3 + \bar{x}_3x_4x_5x_6x_7 \pmod 2$$

$$f_4 = x_1\bar{x}_2x_3 + x_2x_3\bar{x}_4 + x_2x_3x_7 + \bar{x}_2\bar{x}_3x_5x_6 \pmod 2$$

$$f_5 = x_1\bar{x}_2x_3 + \bar{x}_2\bar{x}_3x_6x_7 + x_2x_3\bar{x}_5 + x_2x_4x_5 \pmod 2$$

$$f_6 = x_1\bar{x}_2x_3 + x_2\bar{x}_3x_4x_5 + x_2x_3x_6x_7 \pmod 2$$

$$f_7 = x_1\bar{x}_2x_3 + x_2\bar{x}_4x_6 + x_2x_4x_5 + \bar{x}_2\bar{x}_3x_7 \pmod 2$$

$$f_8 = x_1x_2x_3 + x_2\bar{x}_3x_7 + \bar{x}_2(\bar{x}_3x_6 + \bar{x}_3\bar{x}_4 + x_5x_6 + x_4\bar{x}_5) \pmod 2$$

$$f_9 = x_1\bar{x}_2x_3 + \bar{x}_2\bar{x}_3x_7 + x_2x_3\bar{x}_4x_5 + x_2x_4x_6 \pmod 2$$

$$f_{10} = x_1\bar{x}_2x_3 + \bar{x}_2\bar{x}_3x_7 + x_2x_3\bar{x}_4x_5 + x_2x_3x_4x_6 \pmod 2$$

$$f_{11} = x_1\bar{x}_2\bar{x}_3 + x_2\bar{x}_3\bar{x}_4x_5 + x_2x_4x_6 + x_3\bar{x}_4x_7 \pmod 2$$

$$f_{12} = x_1\bar{x}_2\bar{x}_3 + \bar{x}_2x_3x_4x_5 + x_2\bar{x}_4x_7 + x_2x_3x_6x_4 \pmod 2$$

$$f_{13} = x_1\bar{x}_2\bar{x}_3 + x_2x_3x_4\bar{x}_5 + \bar{x}_2x_3\bar{x}_4x_6 + x_2x_4x_5x_6 \pmod 2$$

$$f_{14} = x_1\bar{x}_2\bar{x}_3 + \bar{x}_2x_3x_4\bar{x}_5 + \bar{x}_2x_3\bar{x}_4x_6 + x_2x_5x_7 \pmod 2$$

$$f_{15} = \overline{x}_3(x_1\overline{x}_2 + x_2x_4x_5) + x_3(\overline{x}_2x_5 + \overline{x}_2\overline{x}_7 + x_5x_6 + \overline{x}_6x_7) \pmod 2$$
$$f_{16} = x_1\overline{x}_2\overline{x}_3 + \overline{x}_2x_3\overline{x}_4x_5 + x_2x_3x_4x_6 + \overline{x}_2x_3\overline{x}_5x_7 \pmod 2$$
$$f_{17} = x_1\overline{x}_2x_3 + x_2x_4x_5x_6 + \overline{x}_2\overline{x}_3x_7 \pmod 2$$
$$f_{18} = x_1\overline{x}_2x_3 + \overline{x}_2\overline{x}_3x_7 + x_2x_3\overline{x}_4 + x_2x_4x_5x_6 \pmod 2$$
$$f_{19} = x_1\overline{x}_2x_3 + \overline{x}_2\overline{x}_3x_7 + x_2x_3\overline{x}_5x_6 + x_2x_3x_4x_5 \pmod 2$$
$$f_{20} = x_1x_2x_3 + x_2x_3x_7 + x_2x_3x_4x_5x_6 \pmod 2$$
$$f_{21} = x_1\overline{x}_2x_3 + \overline{x}_2\overline{x}_3x_7 + x_2x_4x_7 + x_2\overline{x}_4x_5x_6 \pmod 2$$
$$f_{22} = x_1\overline{x}_2x_3 + x_2\overline{x}_4x_7 + x_2x_3x_4x_5x_6 \pmod 2$$
$$f_{23} = x_1\overline{x}_2x_3 + x_2\overline{x}_4x_5x_6 + \overline{x}_2\overline{x}_3x_7 + x_2x_4x_7 \pmod 2$$
$$f_{24} = x_1x_2x_3 + \overline{x}_3(\overline{x}_6x_7 + \overline{x}_6\,\overline{x_4x_5} + x_2x_7 + \overline{x}_2x_4x_5) \pmod 2$$
$$f_{25} = x_1x_2\overline{x}_3 + \overline{x}_2\overline{x}_3x_4 + x_3\overline{x}_4x_5x_6 + x_3x_4x_7 \pmod 2$$
$$f_{26} = x_1\overline{x}_2\overline{x}_3 + x_2\overline{x}_4x_5x_6 + x_2\overline{x}_3x_4 + x_3x_4x_7 \pmod 2$$
$$f_{27} = x_1\overline{x}_2\overline{x}_3 + x_2\overline{x}_4x_5x_6 + x_2\overline{x}_3x_4x_5 + x_3x_4x_7 \pmod 2$$
$$f_{28} = x_1\overline{x}_2x_3 + \overline{x}_2\overline{x}_3x_4x_5 + x_2\overline{x}_4x_5x_6 + x_2x_4x_7 \pmod 2$$
$$f_{29} = x_1\overline{x}_2\overline{x}_3 + x_2\overline{x}_3x_4x_5x_6 + x_3x_4x_7 \pmod 2$$
$$f_{30} = x_1\overline{x}_2\overline{x}_3 + x_2\overline{x}_3x_4x_5 + x_3x_4x_6x_7 \pmod 2$$
$$f_{31} = x_1x_2\overline{x}_3 + \overline{x}_2\overline{x}_3x_4x_7 + x_2x_3\overline{x}_4x_5 + x_3x_4x_5x_6 \pmod 2$$
$$f_{32} = x_1\overline{x}_2x_3 + \overline{x}_2\overline{x}_3x_4x_7 + x_2x_4x_5\overline{x}_6 + x_2x_3x_5x_6 \pmod 2$$
$$f_{33} = x_2(x_1\overline{x}_3 + x_3x_4) + \overline{x}_2(\overline{x}_4x_5 + \overline{x}_4\overline{x}_7 + x_5x_6 + \overline{x}_6x_7) \pmod 2$$
$$f_{34} = x_3(x_1\overline{x}_2 + x_2x_4x_5) + \overline{x}_3(\overline{x}_4x_5 + \overline{x}_4x_7 + x_5x_6 + \overline{x}_6x_7) \pmod 2$$
$$f_{35} = x_1\overline{x}_2x_3 + \overline{x}_3x_4x_7 + \overline{x}_3x_6\overline{x}_7 + x_2x_3x_4\overline{x}_5 + x_2x_3x_5x_6 \pmod 2$$
$$f_{36} = x_1\overline{x}_2x_3 + \overline{x}_1\overline{x}_3x_4x_5 + x_2\overline{x}_3\overline{x}_5\overline{x}_6 + x_2x_3x_6\overline{x}_7 + x_2x_3x_4x_7 \pmod 2$$
$$f_{37} = x_1\overline{x}_2\overline{x}_3 + \overline{x}_2x_3x_6 + x_2\overline{x}_4x_5x_6 + x_2x_4x_5x_7 \pmod 2$$
$$f_{38} = x_1x_2x_3 + \overline{x}_3(x_2\overline{x}_7 + \overline{x}_7\,\overline{x_5x_6} + x_2x_4 + \overline{x}_4x_5x_6) \pmod 2$$

$f_{39} = x_1\bar{x}_2\bar{x}_3 + \bar{x}_2 x_3 x_4 + x_2\bar{x}_4 x_5 x_6 + x_2 x_4 x_5 x_7 \pmod 2$

$f_{41} = x_1\bar{x}_2\bar{x}_3 + \bar{x}_2 x_3 x_4 x_7 + x_2\bar{x}_4 x_5 x_6 + x_2 x_4 x_5 x_7 \pmod 2$

$f_{42} = x_1\bar{x}_2\bar{x}_3 + x_2\bar{x}_4 x_5 x_6 + x_2 x_4 x_5\bar{x}_7 + x_2 x_3 x_4 x_7 \pmod 2$

$f_{43} = x_1\bar{x}_2 x_3 + \bar{x}_2\bar{x}_3 x_4 x_5 + x_2\bar{x}_4 x_5 x_6 + x_2 x_4 x_5 x_7 \pmod 2$

$f_{44} = x_1\bar{x}_2\bar{x}_3 + x_2 x_3\bar{x}_4 x_5 + x_2 x_4\bar{x}_5 x_6 + x_3 x_4 x_5 x_7 \pmod 2$

$f_{45} = x_1 x_2\bar{x}_3 + x_2 x_3 x_4 x_5 x_6 + x_3 x_4 x_5 x_7 \pmod 2$

$f_{46} = x_1 x_2\bar{x}_3 + \bar{x}_2 x_3 x_4 x_5 x_6 + x_2 x_3 x_4 x_5 x_7 \pmod 2$

$f_{47} = x_1\bar{x}_2 x_3 + x_2 x_4\bar{x}_5 x_6 x_7 + x_2 x_3 x_5 \pmod 2$

$f_{48} = x_1\bar{x}_2 x_3 + x_2 x_3\bar{x}_5 x_7 + x_2 x_4 x_5 x_6 x_7 \pmod 2$

$f_{49} = x_1\bar{x}_2 x_3 + \bar{x}_2\bar{x}_3 x_5 x_6 x_7 + x_2 x_4 x_5 x_6 \pmod 2$

$f_{50} = x_1\bar{x}_2 x_3 + x_2 x_3 x_4 x_5 x_6 x_7 \pmod 2$

В заключение хочу выразить благодарность доц.К.Чимеву, под руководством которого была выполнена эта работа.

## Л И Т Е Р А Т У Р А

1. Яблонский С.В. Функциональные построения в к-значной логике. Труды МИАН им.Стеклова, т.51,1958,5-122.
2. Чимев К.Н.,Г.А.Кутирков. Върху променливите от първи порядък за функциите от многозначната логика. Год.на ВТУЗ, Математика, т.V,кн.3,1968/69.
3. Чимев К.Н. Отделими двойки и силно съществени променливи на функциите на три, четири и пет аргумента. Год.на ВТУЗ,Математика,т.VII,кн.3, 1971.
4. Чимев К.Н. Върху отделимите двойки на функциите. Год. на ВТУЗ,Математика,т.VIII,кн.3,1972.
5. Чимев К.Н. Върху променливите от втори порядък. Год. на ВТУЗ, Математика,т.VIII,кн.2,1972.

6. Чимев К.Н Върху инвариантността на отделимите двойки на функциите. Год.на ВТУЗ,Математика, т.VIII, кн.1, 1972.
7. Чимев К.Н. Отделими двойки на функциите на шест аргумента. Год.на ВТУЗ,Математика, т.XII,кн.2, 1976.
8. Чимев К.Н. Върху някои дискретни функции. Год.на ВТУЗ, Математика,т.VI,кн.2,1970.
9. Чимев К.Н. Върху един клас функции. Год.на ВТУЗ, Математика,т.VI,кн.3,1970.
10. Брейтбарт Ю.Л. О существенных переменных функции алгебры логики. Доклады АН СССР, № 1, 1967.

## ON THE SEPARATE PAIRS OF A CLASS OF FUNCTION

I.D. Gjudjenov

Abstract

The problem of the separate pairs of variables of Boolean functions essentially depending on seven variables and having variables of first or second degree, is treated.

MTA SZTAKI Tanulmányok 147/1983 61-71

# ON THE STIRLING NUMBERS OF THE SECOND KIND

T. Lengyel

Computer and Automation Institute
Hungarian Academy of Sciences

Several authors investigate the asymptotic behaviour of the Stirling numbers $S(n,k)$ of the second kind. Up to now, there is no common asymptotic estimation of $S(n,n-k)$ for each $k: 1 \leq k \leq n$. In this paper we discuss the behaviour of Hsu's expansion for $S(n,n-k)$ as $k \to \infty$. We extend his expansion and prove that it will converge for "relatively small" $k$, as $k \leq \frac{c \log n}{\log \log n}$ $0<c<1$.

## 1. Introduction

Hsu [2] has given the asymptotic expansion, for any fixed $k$

$$(1.1) \qquad S(n+k,n) = \frac{n^{2k}}{2^k k!} \{1+ \sum_{m=1}^{\infty} \binom{k}{2m} W(m,n)\}\{1+ \sum_{m=1}^{k} \frac{S_m(k)}{n^m}\} =$$

$$= \frac{n^{2k}}{2^k k!} \{1+ \sum_{m=1}^{t} \binom{k}{2m} W(m,n)+O(n^{-t-1})\}\{1+ \sum_{m=1}^{t} \frac{S_m(k)}{n^m} +O(n^{-t-1})\}$$

where

$$W(m,n) = B_{2m}^{-n}(-\tfrac{1}{2}n) \Big/ (\tfrac{1}{2}n)^{2m} ,$$

$B_{2m}^{-n}(x)$ being a so-called Bernoulli polynomial of negative order. The sum of all $\binom{n}{k}$ products of k different numbers taken from the set $\{1,2,\ldots,n\}$ is denoted by $S_k(n)$.

He did not investigate the error terms standing within the brackets as k tends to infinity.

Bleick and Wang [1] showed by numerical evidence that this extimation does not work for arbitraly large k. They have given on asymptotic expansion for $S(n,n-k)$ and proved its convergence for k: $n-k \leq Kn^{2/3}$ where K is a positive constant.

We prove that some modification of Hsu's expansion for $S(n,n-k)$ converges for $k \leq \frac{c \log n}{\log \log n}$ :

$$(1.2) \qquad S(n+k,n) = \frac{n^{2k}}{2^k k!}\{1+\sum_{m=1}^{t}\binom{k}{2m}W(m,n)+O(\binom{k}{2t+2}W(t+1,n)\}\cdot$$

$$\cdot\{1+\sum_{m=1}^{t}\frac{S_m(k)}{n^m}+O(\frac{S_{t+1}(k)}{n^{t+1}})\}^{-}$$

In Section 2 we discuss the error terms in the expansion (1.1) as $k \to \infty$.

Theorem and Lemma 1 discuss the term involvi g . noulli polinomials. Lemma 2, Lemma 3, Lemma 4, Lemma 5 and Lemma 6 concern with the remaining error term as a function of m. These five lemmas take care of the values of m chosen from the interval $[1,k]$ and they prove that $S_m(k)/n^m$ decrease rapidly with increasing m, as $n \to \infty$. Obviously,

$S_m(k) = 0$ for $m > k$.

## 2. Lemmas

Kimball has given the following theorem ([3], Theorem 2)

<u>Theorem.</u> For all positive integral values of m and n the function $B_{2m}^{-n}(-\frac{1}{2}n)$ satisfies the relation

$$B_{2m}^{-n}(-\tfrac{1}{2}n) \leq \frac{n(n-1)\ \ldots\ r}{(n+2m)(n+2m-1)\ldots(r+2m)}\ (\tfrac{1}{2}n)^{2m}$$

where $r = [(n+1)/2]$.

Easy computation shows that

$$B_{2m}^{-n}(-\tfrac{1}{2}n) \leq \left(1 - \frac{2m}{n+2m}\right)^{r+1}\left(1 - \frac{2m}{n(n+2m-1)}\right)^{r+1}\left(\tfrac{1}{2}n\right)^{2m}$$

This gives for $m = o(n^{1/2})$ that $(K_1 < \infty)$

$$W(m,n) \leq K_1\, e^{-\frac{2m}{n+2m}(r+1)}(1+o(n^{-1/2})) \leq e^{-m}\, O(1).$$

We consider the terms in the (finite) sum

$$\sum_{m=1}^{\infty}\binom{k}{2m}W(m,n). \tag{2.1}$$

For $k/3 < m$ we can easily see that these terms are rapidly decreasing thus the test ratio for convergence holds.

For small values of m we can extend [3], Corollary 1 on

$$\lim_{n\to\infty}\frac{B_{2m}^{-n}(-\frac{1}{2}n)}{n^m}. \tag{2.2}$$

These two considerations imply

Lemma 1.

$$1 + \sum_{m=1}^{\infty}\binom{k}{2m}W(m,n) = 1 + \sum_{m=1}^{t}\binom{k}{2m}W(m,n) + O\left(\binom{k}{2t+2}W(t+1,n)\right) \tag{2.3}$$

as $n \to \infty$ and $k \leq \frac{c \log n}{\log \log n}$

One can observe that

$$S_m(k) = s(k+1, k+1-m)$$

where $s(n,k)$ denotes the Stirling number of the first kind.

We suppose that $k = o(n^{1/2})$ as $n \to \infty$.

Lemma 2.

(2.4) $$S_m(k) = \frac{1}{m!}\left(\frac{k^2}{2}\right)^m (1+o(1))$$

holds as $k \to \infty$ and $m = o(k^{1/2})$.

Proof: See Sachkov [4] (III.5, (5.5) Moser and Wyman's asymptotical result on the Stirling number of the first kind).

This implies that $S_m(k)/n^m$ decrease exponentially with increasing m, as $m = o(k^{1/2})$ and $n \to \infty$.

Now we suppose that

$$(k+1)^{1/4} \leq m.$$

Moser and Wyman have given the following result on $s(n,m)$, the Stirling numbers of the first kind (see [4] III.5, (5.2)). Suppose that

$$0 < h(n) \leq m \leq n-g(n) \tag{2.5}$$

hold for each m, where $g(n) > 0$ and $g(n) \geq cn^{\alpha}(c > 0, 0 < \alpha < 1)$ suppose furthermore that $h(n)$ is an arbitrary diverging function, i.e. $h(n) \to \infty$ as $n \to \infty$. Then

$$s(n,m) = \frac{\Gamma(n+R)}{R^m \Gamma(R)\sqrt{2\pi H}} (1 + o(1)) \tag{2.6}$$

as $n \to \infty$, where R is the unique solution of the equation

$$\sum_{h=o}^{n-1} R/(R+h) = m \qquad (R = R(n,m)) \tag{2.7}$$

and

$$H = m - \sum_{h=o}^{n-1} R^2/(R+h) \qquad (H = H(n,m)). \tag{2.8}$$

Easy computations shows that both R and H are bounded by some polinomial function of n and m.

In our case we obtain from (2.6) that

$$s(k+1, k+1-m) = \frac{\Gamma(k+1+R)}{R^{k+1-m} \Gamma(R)\sqrt{2\pi H}}(1 + o(1)) \tag{2.9}$$

as $k \to \infty$, , and $g(k+1) \leq m \leq k+1-h(k+1)$. We can choose $g(h) = k^{1/4}$ and $h(k) = (\ln k)^{1/2}$. From (2.7) we have

$$R \ln(1 + \frac{1 + k}{R}) = k + 1 - m.$$

Replacing R by $(k + 1)d$ $(d = d(k,m))$ we obtain the equation to be solve

$$d \ln(1 + \frac{1}{d}) = 1 - \frac{m}{k + 1} . \tag{2.10}$$

We discuss the solution d of (2.10) in five intervals covering values of $m : 1 \leq m \leq k$. By simple consideration we have the following three lemmas.

Comparing to Lemma 2 the next lemma extends the range of those values of m, for which $S_m(k)/n^m$ rapidly decrease with increasing m.

Lemma 3. For $(k+1)^{1/4} \leq m = o(k)$ we have

$$d = \frac{k + 1}{2m} (1 + o(\frac{1}{m^\beta})) \qquad (\beta \geq 0).$$

This implies that $S_m(k)/n^m$ decrease exponentially with increasing m, as $n \to \infty$.

Lemma 4. For m:

$$1 - \varepsilon_1 \leq m/k \leq 1 - \varepsilon_2 \qquad (0 < \varepsilon_2 < \varepsilon_1 < 1)$$

we have

$$k_1 \leq d \leq k_2 \qquad (0 < k_1 \leq k_2 < \infty)$$

Obviously $S_m(k)/n^m$ decrease exponentially.

Lemma 5. Let us suppose that $\frac{m}{k} \to 1$ and $m \leq k + 1 - (\ln k)^{1/2}$. Then

$$d \sim -\frac{1 - \frac{m}{k+1}}{\log\left(1 - \frac{m}{k+1}\right)}$$

as $k \to \infty$.

The exponentially rapid convergence follows again.

Lemma 6. Let us suppose that

$$k + 1 \geq m \geq k + 1 - (\ln k)^{1/2}$$

We have

$$s(k+1, k+1-m) = \frac{k!}{(k-m)!}(\ln(k+1)+C)^{k-m}(1+o(1))$$

as $k \to \infty$, where $C=0.5772...$

Lemma 6 immediately follows from Jordan's formula (see [4], III.5, (5.1)). The rate of convergence is exponential (as m increases and $n \to \infty$). From Lemma 2-6 follows that for $k = o(n^{1/2})$

$$(2.11) \qquad \sum_{m=1}^{k} \frac{S_m(k)}{n^m} = \sum_{m=1}^{t} \frac{S_m(k)}{n^m} + O\left(\frac{S_{t+1}(k)}{n^{t+1}}\right)$$

as $n \to \infty$.

The equations (2.3) and (2.11) guarantee that (1.2) holds.

This implies that

$$S(n+k,n) = \frac{n^2 k}{2^k k!}\{1 + \frac{f_1(k)}{n} + \frac{f_2(k)}{n^2} + \ldots + \frac{f_t(k)}{n^t} +$$

$$+ O(\frac{S_{t+1}(k)}{n^{t+1}}) + O(\binom{k}{2t+2} W(t+1,n))\}.$$

as $n \to \infty$ and $k \leq \frac{c \log n}{\log \log n}$ where $f_i(k)$ are polinomials of 2i-degree. We find that

$$f_1(k) = \frac{1}{3}(2k^2 + 2)$$

$$f_2(k) = \frac{1}{18}(4k^4 - k^2 - 3k)$$

$$\vdots$$

etc.

References

1 Bleick, W.E. - Wang, P.C.C., Asymptotics of Stirling numbers of the second kind, Proceedings of the Amer. Math.Soc., 42,/1974/, pp. 575-58o.

2 Hsu, L.C., Note on an asymptotic expansion of the n-th difference of zero, Ann. Math. Statist, 19, /1948/, pp. 273-277.

3 Kimball, B.F., The application of Bernoulli polinomials of negative order to differencing, Amer.Math. Jour., 55, /1933/, pp. 399-416.

4 Sachkov, V.N., Combinatorial methods in discrete mathematics, 1977, Nauka, Moscow /in Russian/

# О ЧИСЛАХ СТИРЛИНГА ВТОРОГО РОДА

Т. Лендьел

## Резюме

В статье изучается асимптотическое поведение чисел Стирлинга второго рода $S(n,n-k)$, где $k \to \infty$. Доказывается, что разложение Хсу сходится при $k \le \frac{c \log n}{\log \log n}$ .

MTA SZTAKI Tanulmányok 147/1983 73-81

# CALCULATION OF COMMUTATOR IDENTITIES IN ALTERNATING GROUPS ON COMPUTERS

D. Nikolova

Institute of Mathematics with Computer Center,
Bulgarian Academy of Sciences

In recent papers of the author the following group identities: $[x_r y]=[x_s y]$ , $r<s$ , have been examined. For example, nilpotent groups of nilpotency class $\leq c$ , Engel groups of Engel class $\leq m$ , finite groups, locally finite varieties of groups satisfy these laws. It was shown [2] how these identities influence the structure of finitely generated solvable groups. With regard to this, one more example of varieties of groups whose laws are finitely based was given. Some particular cases of identities of this type turned out to be equivalent to the commutative law or to the property of nilpotency in some classes of groups. The minimal identities of this type were computed in some groups of small order, as well as (see [3] ) in the varieties of type $\mathfrak{A}_{p^s}\mathfrak{A}_{p^t}$ , where $p$ is a prime number and $\mathfrak{A}_k\mathfrak{A}_l$ , where $(k,l)=1$ ($\mathfrak{A}_k$ is the variety of all abelian groups of exponent $k$ ). The main object of this note is to define algorithms for the computation of the minimal identities of type $[x_r y]=[x_s y]$ in each group of the infinite series of simple alternating groups and the implementation of these algorithms on computers. The more that one aspect which seems to have been little considered is that of the laws which finite nonabelian simple groups satisfy. Two FORTRAN programs were composed - the first one computes the

law in $A_5$ and the other is intended for the general case $A_n$. They differ in their strategy for storing the elements of $A_n$ in the computer. Some of the algorithms employed by the programs are presented in the talk. Programs of the kind described here serve as examples of group-theoretical programs capable of producing mathematical results which otherwise could not be obtained easily.

1. Introduction

We set as usual:

$S_n$ : the symmetric group of degree $n$ ,

$A_n$ : the alternating group of degree $n$ ,

$x^y = yxy^{-1}$ , $[x,y] = xyx^{-1}y^{-1}$,

$[x,_r y] = [x, \underbrace{y,y,\dots,y}_{r \text{ entries}}] = [[x, \underbrace{y,y,\dots,y}_{r-1 \text{ entries}}], y]$

(following Gruenberg).

In a recent paper [2] we examined groups which satisfy the law:

$$[x,_r y] = [x,_s y] \quad , \quad r < s \qquad (1)$$

We order the identities of the type (1), which are satisfied in a given group $G$ in the following way:

$$[x,_{m_1} y] = [x,_{n_1} y] < [x,_{m_2} y] = [x,_{n_2} y] \quad \text{iff} \quad (m_1,n_1) < (m_2,n_2)$$

lexicographically. It is interesting to know the minimal identity of the type in a certain group, because of Theorem 1 below (see [2] ):

Theorem 1. Let $G$ be a group which satisfies a law of the type (1) and $[x,_{m_0} y] = [x,_{n_0} y]$ be the minimal identity of this type in $G$ , $d = n_0 - m_0$ . Then $G$ satisfies the law $[x,_{m'} y] = [x,_{n'} y]$ iff $m_0 \leq m'$ and $d \mid (n'-m')$.

We shall demonstrate the computation of this minimal identity in simple alternating groups $A_n$, for $n \geq 5$.

2. The law in $A_5$.

In this section a particular case is examined, i.e. the search for the minimal identity of the type (1) in the group $A_5$.

Theorem 1 gives us a practical method for finding out the integers $m_0, n_0$ of the minimal identity of the type (1) in a finite group $G$:

Algorithm 1. A finite group $G$ is given. It is desired to produce integers $m_0, n_0$ such that $[x, m_0 y] = [x, n_0 y]$ is a law in $G$ and such that $(m_0, n_0)$ is the minimal lexicographical ordered pair.

(a) Examine all $l = (|G|-1)(|G|-2)$ ordered pairs of elements in $G \setminus \{1\}$, their corresponding minimal equalities of the type (1) and differences:

$$g_1, h_1 \in G ; [g_1, m' h_1] = [g_1, n' h_1] ; d' = n' - m'$$

$$g_2, h_2 \in G ; [g_2, m'' h_2] = [g_2, n'' h_2] ; d'' = n'' - m''$$

.....

$$g_l, h_l \in G ; [g_l, m^{(l)} h_l] = [g_l, n^{(l)} h_l] ; d^{(l)} = n^{(l)} - m^{(l)}.$$

(b) Find $m_0 = \max(m', m'', \ldots, m^{(l)})$.

(c) Determine the smallest common multiple $d$ of the numbers $d', d'', \ldots, d^{(l)}$.

(d) Set $n_0 = m_0 + d$.

Evidently $(m_0, n_0)$ is the pair we look for and it is uniquely defined.

We shall make some remarks about the FORTRAN program implementation:

(1) It is necessary to discuss the method to be used

for storing the permutations in the computer. We write the permutation $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ \pi_1 & \pi_2 & \pi_3 & \pi_4 & \pi_5 \end{pmatrix}$ as an ordered 5-tuple $(\pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$. We produce the elements of $S_5$ in lexicographical order using a block-schema. This is implemented by the subroutine DATA. As $S_5$ is not a big group, we write the elements in the two-size massif of variables $S_5(5, 120)$

(2) Given a permutation $\pi \in S_5$ it should be easy to determine whether or not $\pi$ is in $A_5$. This is implemented by the subroutine ALT following the algorithm:

Algorithm 2 for separating the even from the odd permutations.

(a) Set $A=0$, $i=1$ and examine $\pi^{(i)} \in S_5$ We write in the variable $A$ the number of inversions of the given permutation.

(b) Calculate the number of inversions of $\pi^{(i)}$ and set $A$ equal to that number.

(c) If 2 divides $A$, then write $\pi^{(i)}$ in the massif $Y$. Otherwise go to step (d).

(d) Set $i := i+1$ and go to step (b).

We collect the elements of the alternating group $A_5$ in the two-size massif $Y(5, 60)$.

(3) To reduce the number of computations of step (a) from the Algorithm 1 we note that it is sufficient to search the minimal equations of the type (1) for the ordered pairs $(x, y)$ of elements, where $y$ runs through the elements of $A_5$ and $x$ - only through representatives of the conjugacy classes of $A_5$. In fact:

$$[x, y] = a^{-1}[x^a, y^a]a \quad , \quad \forall a \in A_5.$$

Then $[x, y] = [x, y] \Leftrightarrow a^{-1}[x^a, y^a]a = a^{-1}[x^a, y^a]a$

$\Leftrightarrow [x^a, {}_m y^a] = [x^a, {}_n y^a]$, $\forall a \in A_5$.

We choose the representatives $x_1 = (123)$, $x_2 = (12)(34)$, $x_3 = (12345)$, $x_4 = (12354)$ and we store them in the two-size massif $X(5,4)$.

(4) Algorithms for the handling, i.e. comparison, multiplication, inversion of the group elements must be defined. We compare two elements of the group $A_5$, as we write them as 5-digit numbers.

Algorithm 3 for multiplication of permutations of degree 5 :

Two permutations are given: $(i_1, i_2, i_3, i_4, i_5)$ and $(j_1, j_2, j_3, j_4, j_5)$. As a result of their multiplication we get the permutation $(k_1, k_2, k_3, k_4, k_5)$ by the following procedure:

If $i_t = s$, then $k_t = j_s$, $s = 1 \div 5$, $t = 1 \div 5$.

Algorithm 4 for inverting permutations of degree 5:

The permutation $\pi = (\pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$ is given. Set $p = (p_1, p_2, p_3, p_4, p_5) = \pi^{-1}$. We get $p$ in the following way:

if $\pi_l = k$, then $p_k = l$, $k = 1 \div 5$, $l = 1 \div 5$.

All these algorithms are implemented by the main subroutine DANI2 on the way of finding out the law. The inverse elements we write in the one-size massif $X1(5)$, $Y1(5)$ every time we need them.

(5) The computing time for this task on the machine EC 1040 is 4,07 minutes. Finding out the minimal identity of the type (1) in $A_5$ could be done routinely by a labourious hand computation. As normal subgroups do not exist in the group we do not know the order of an element which is the product of two elements belonging to different subgroups of $A_5$. So it is not sufficient to know the laws in each subgroup of $A_5$,

because it may happen that $x, y \in A_5$ generate the whole group. Moreover we make use of this program to extend the calculations to greater $n$.

We obtained that the minimal identity of the type (1) in the group $A_5$ is $[x_{1,3}y] = [x_{1,63}y]$.

3. A program for the computation of identities in alternating groups $A_n$, $n \geq 5$.

In this section the computations necessary for finding out the minimal identity of type (1) in the finite simple group $A_n$ for $n=5$ will be generalized. The program implementation of this task differs from that of section 2 mainly in the strategy for storing the permutations in the computer. The condition to keep all elements in a (fast-access) store is a serious restriction for great $n$. We need an efficient method with respect to the storage for generating each permutation from the previous one every time it is necessary in the program. Algorithm 5 (see [1]) implements such a method. Like every algorithm for systematic generation of elements it comprises three components: A choice of initial configuration, a transformation of the object to the following one and a condition for ending the process.

Algorithm 5 for the generation of permutations in minimal alteration order:

For $i = 1$ to $n$ do $\begin{cases} \pi_i \leftarrow p_i \leftarrow i \\ d_i \leftarrow -1 \end{cases}$

$d_1 \leftarrow 0$; $\pi_0 \leftarrow \pi_{n+1} \leftarrow m \leftarrow n+1$

while $m \neq 1$ do

deduce $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$

$m \leftarrow n$

while $\pi_{p_m + d_m} > m$ do $\begin{cases} d_m \leftarrow -d_m \\ m \leftarrow m-1 \end{cases}$

$\pi_{p_m} \leftrightarrow \pi_{p_m + d_m}$

[[in that moment $\pi_{p_m + d_m} = m$]]

$p_{\pi_{p_m}} \leftrightarrow p_m$

Some minor modifications have been made for FORTRAN program implementation of this algorithm. Let us make some additional remarks:

(1) As permutations differ only by a transposition on every step, the even permutations alternate with odd ones, so there is no need of special algorithm for generating the elements of $A_n$.

(2) The correctness of Algorithm 5 is proved in [1] by simple induction on $n$.

(3) Algorithm 5 is linear, i.e. the computing time for generating all the elements of the set is proportional to its size. Such algorithms have the best asymptotic efficiency. Moreover, as each permutation on our list differs from its predecessor only by a transposition of two adjacent entries, that algorithm is of minimal alteration. Hence Algorithm 5 is one of the most efficient algorithms for generating permutations.

(4) Algorithm 5 terminated in a finite number of steps.

(5) In this program in order to find out the minimal equalities of type (1) for the ordered peirs of elements $(x,y)$, $x, y$ range over all the elements of $A_n$ because it serves no purpose to determine the representatives of the conjugacy classes, written as nth dimensional vectors. Either they can be obtained after an individual hand computation or if one wants to produce them automatically, one has to present each permutation as a product of independent cycles. For the latter purpose we need methods for introducing round brackets in a given $n$-tuple $\pi$.

(6) The main algorithm works for determining the minimal identities of the type (1) in the symmetric groups $S_n$.

The search of minimal identities of the type (1) in

alternating groups $A_n$ with computer is implemented by the main subroutine DANI 3.

The other remarks we did in section 2 about $A_5$ apply to $A_n$, for $n>5$.

The necessary computations were done by a computer for the first few $n$. For the alternating group $A_6$ of degree 6, for example, we obtained the law: $[x_{,14}\, y] = [x_{,124}\, y]$.

REFERENCES

1. E. Reingold, J. Nievergelt, N.Deo. Combinatorial Algorithms. Theory and Practice, Prentice-Hall Inc., Englewood Cliffs, New Jersey 07632, 1977.
2. Д.В.Николова. Об одном классе тождеств в группах. Сердика (в печати).
3. Д.В.Николова. Тождества в метабелевых многообразиях групп $\mathcal{U}_k \mathcal{U}_\ell$. Сердика (в печати).

# ВЫЧИСЛЕНИЕ НА ЭВМ ОДНОГО КОММУТАТОРНОГО ТОЖДЕСТВА В ЗНАКОПЕРЕМЕННЫХ ГРУППАХ

Даниела Николова

**Резюме**

Тождества вида $[x,_r y]=[x,_s y], r<s$, дают некоторые структурные характеристики групп, как например, коммутативность, нильпотентность, энгелевость. Эти тождества выполняются в каждой конечной группе. В работах автора показано каким образом их выполнение отражается на строении разрешимых групп с конечным числом образующих (см. [2] ) и каким будет минимальное тождество этого вида в многообразиях $\mathcal{U}_k \mathcal{U}_l$ (см. [3] ). Целью этой работы является составление алгоритмов вычисления минимального тождества вида $[x,_r y]=[x,_s y]$ в каждой группе из бесконечной серии простых знакопеременных групп и их реализация на ЭВМ. В статье описаны необходимые алгоритмы и привидены две программы на языке FORTRAN-первая вычисляет тождество в $A_5$, а другая предназначена для общего случая $A_n$. Они отличаются между собой стратегией запоминания элементов $A_n$ в машине. Эти программы служат в качестве примеров теоретико-групповых программ, способных произвести математические результаты, которые трудно получить иным путем. Даются результаты вычислений для нескольких первых значений числа $n$.

MTA SZTAKI Tanulmányok 147/1983 83-99

# UNIFIED APPROACH TO THE APPROXIMATION ON FINITE POINT SETS

B. Uhrin

Computer and Automation Institute
Hungarian Academy of Sciences

## INTRODUCTION

Let $X$ be an arbitrary set having at least $n+1$ points, $n \geq 1$. Denote by $F(X)$ the set of real valued functions defined on $X$. Let $\varphi_1, \varphi_2, \ldots, \varphi_n$ be $n$ functions from $F(X)$ which are linearly independent on $X$. Let us denote by $\Phi$ the subspace of $F(X)$ spanned by $\varphi_i$-s. The elements of $\Phi$ are called generalized /g/-polynomials.
Let $f \in F(X) \setminus \Phi$, $Q$ be a subset of $X$ and denote by $\vartheta$ any g-polynomial such that $\vartheta(x) \not\equiv 0$ on $Q$.

*Definition 1*. We say that $\vartheta$ is the generalized /g/-juxtapolynomial of $f$ on $Q$ if there is no $\varphi \in \Phi$ such that

$$\begin{aligned} &\varphi(x) = 0 \quad \text{if} \quad x \in Q \quad \text{and} \quad f(x) = 0, \\ &\varphi(x) > 0 \quad \text{if} \quad x \in Q \quad \text{and} \quad f(x) > 0, \\ &\varphi(x) < 0 \quad \text{if} \quad x \in Q \quad \text{and} \quad f(x) < 0. \end{aligned} \tag{0.1}$$

*Definition 2.* We say that $\varphi \in \Phi$, $\varphi(x) \not\equiv 0$ on $Q$, is a generalized /g/-juxtapolynomial of $f$ on $Q$ if $\vartheta$ is the g-juxtapolynomial of $f-\varphi$ on $Q$.

This concept is a natural generalization of "polynomials nearest to $f$ in a monotone distance function" due to Fejér [1], "extremal polynomials" of Fekete and von Neumann [2], "nearest polynomials" of Fekete [3] or "/algebraic/ juxtapolynomials" of Motzkin and Walsh [4].

For some results concerning g-juxtapolynomials on $Q \subset X = C$ /the complex plane/ we refer to Shisha [5] or Marusciac [6]. A detailed discussion of continuous g-juxtapolynomials on $Q \subset X = [0,1]$ can be found in Rice [7, pp. 260-305]. If $Q$ is a finite subset of $X$ then any function $\varphi \in \Phi$ that is nearest to $f$ in a distance function $\delta(g)$: $F(X) \to R_+^1$ which is monotone on $Q$ /in the sense of Fejér [1]/, is a g-juxtapolynomial of $f$ on $Q$.

*Remark:* We recall that Fejér called $\delta(g)$ monotone on $Q$ if

$$g(x) \not\equiv 0 \text{ on } Q \Rightarrow \delta(g) > 0$$

and

$$|h(x)| \begin{cases} = 0 & \text{if } x \in Q \text{ and } g(x) = 0 \\ < |g(x)| & \text{if } x \in Q \text{ and } g(x) \neq 0 \end{cases} \Rightarrow \delta(h) < \delta(g).$$

Examples of monotone distance functions:

$$(0.2) \qquad \sum_{x \in Q} w(x)|g(x)|^p , \qquad 0 < p < \infty ,$$

$$(0.3) \qquad \max_{x \in Q} w(x)|g(x)| ,$$

where $w(x) > 0$, $x \in Q$.

We see that the set of g-juxtapolynomials is sufficiently large to include many "best approximating polynomials".
The paper has four sections.
In Section 1 we recall some classical results. In Section 2 we give a characterization of q-juxtapolynomials in the case when $X$ is totally ordered, $Q$ is finite and $\Phi$ is a Chebyshev system of order $n$. Section 3 contains the extension of results to weak Chebyshev systems. Finally in Section 4 we give some remarks and compare our results with known ones.

## 1. Three classical results

First we prove the following

*Theorem* (Fejér [1]).
Let $E$ be a compact subset of the complex plane $C$ having cardinality at least $n$. Let $p^*(z) = z^n + a^*_{n-1}z^{n-1}+\ldots+ a^*_o$ be a complex monic polynomial of degree $n$ such that for some monotone distance function

$$\delta(p^*) \leq \delta(p) \tag{1.1}$$

holds for all complex monic polynomials $p(z) = z^n + a_{n-1}z^{n-1}+\ldots a_o$ of degree $n$. Then all roots of $p^*$ lie in the convex hull of $E$. □

*Proof:* Let $y^* \in C$ be a root of $p^*$ such that $y^* \notin$ conv $E$. The set conv $E$ is compact convex, hence there is $y \in C$ such that

$$|y-z| < |y^*-z| \quad \text{for all} \quad z \in E. \tag{1.2}$$

The polynomial

$$q(z) = \frac{p^*(z)}{z-y^*}\,(z-y) \tag{1.3}$$

is monic a fulfils the condition

$$|q(z)| \begin{cases} = 0 & \text{if} \quad p^*(z) = 0 \\ < |p^*(z)| & \text{if} \quad p^*(z) \neq 0 \end{cases} \tag{1.4}$$

for all $z \in E$.
Hence $\delta(q) < \delta(p^*)$, a contradiction. ■

Let $E \subset C$ be again a compact set.
Fekete and von Neumann said that a polynomial $q$ (of degree $n$) is "on $E$ nearer to the zero polynomial than the polynomial $p$", if

$$(1.5)\qquad |q(z)| \begin{cases} = 0 & \text{if} \quad p(z) = 0 \\ < |p(z)| & \text{if} \quad p(z) \neq 0 \end{cases}$$

for all $z \in E$.
They called the monic polynomial $p$ extremal on $E$ if there is no monic polynomial $q$ which is nearer on $E$ to 0 than $p$. It is clear from the proof of the Fejér's theorem that it remains true when $p^*$ is an extremal monic polynomial.
In the symmetric case a new statement can be proved.

*Theorem* (Fekete and von Neumann [2]).

Let $E \subset C$ be compact and symmetric about real axis. If $p^*$ is an extremal monic polynomial on E, such that it has all coefficients real, then all roots of $p^*$ lie in the set $\bigcup_{z \in E} G(z)$, where

$$(1.6)\qquad G(z) = \{y \in C: |y - \frac{z+\bar{z}}{2}| \leq |\frac{z-\bar{z}}{2}|\}$$

($\bar{z}$ is the complex conjugate of $z$). □

*Proof:* We first prove that the statement is not empty, i.e. that there is such extremal monic polynomial $p^*$ which has all coefficients real. Indeed, let $p_1 = z^n + a_{n-1}z^{n-1} + \ldots + a_o$ be an polynomial which is extremal on $E$. The symmetricity of $E$ implies that also $p_2 = z^n + \bar{a}_{n-1}z^{n-1} + \ldots + \bar{a}_o$ is an extremal polynomial. But clearly $p^* = \frac{p_1+p_2}{2}$ is also extremal on $E$, and it has real coefficients.
As to the proof of theorem, assume first that E has only finite number of points. Let $a \notin \bigcup_{z \in E} G(z)$ be a root of $p^*$. $p^*$ has real coefficients, hence $\bar{a}$ is also a root of $p^*$. Choose $b \notin \bigcup_{z \in E} G(z)$ such that $b = \lambda a + (1-\lambda)\bar{a}$, $0 < \lambda < 1$. We claim that

$$(1.7)\qquad |(z-b)||(z-\bar{b})| < |z-a|\,|z-\bar{a}| \quad \text{for all} \quad z \in E.$$

It is clear that $|a-z| > |b-z|$, $|a-\bar{z}| > |b-\bar{z}|$ for all $z \in E$, i.e. $|\bar{a}-\bar{z}| > |\bar{b}-\bar{z}|$, $|\bar{a}-z| > |\bar{b}-z|$ for all $z \in E$, implying (1.7).
But (1.7) yields

$$\left| \frac{p^*(z)}{(z-a)(z-\bar{a})} (z-b)(z-\bar{b}) \right| < |p^*(z)| \quad \text{for all} \quad z \in E,$$

and this contradicts to the extremality of $p^*$.
For infinite compact $E$ we have to use the compactness of $E$ to prove the theorem. ∎

Applying this theorem to $E \subset R^1$, we see that all roots of a real monic polynomial $p^*$ which is extremal on $E \subset R^1$ /in the class of all monic real polynomials/ lie in $E$.
A similar result has been proved for real polynomials by Motzkin and Walsh [4]. Denote by $P_{n-1}$ the set of real algebraic polynomials of degree at most $n-1$. They called $p \in P_{n-1}$ as a juxtapolynomial of $f \in F(R^1) \setminus P_{n-1}$ on $Q \subset R^1$ if there is no $q \in P_{n-1}$ such that

$$(1.8) \qquad |f(x)-q(x)| \begin{cases} = 0 & \text{if} \quad f(x) = p(x) \\ < |f(x)-p(x)| & \text{if} \quad f(x) \neq p(x) \end{cases}$$

for all $x \in Q$.

They proved

*Theorem* (Motzkin and Walsh [4]). Let $Q = \{x_1, x_2, \ldots, x_m\} \subset R^1$, $m \geq n+1$, $x_1 < x_2 < \ldots < x_m$. The polynomial $p \in P_{n-1}$ is a juxtapolynomial of $f \in F(R^1) \setminus P_{n-1}$ on $Q$ if and only if there are $n+1$ indices, $1 \leq i_1 < i_2 < \ldots < i_{n+1} \leq m$, and $|\varepsilon| = 1$ such that

$$(1.9) \qquad \varepsilon(-1)^k \left(f(x_{i_k}) - p(x_{i_k})\right) \geq 0 \quad \text{for} \quad k = 1,2,\ldots,n+1. \quad \square$$

This theorem is a simple consequence of a much more general theorem proved in the next section. Applying this theorem to $f(x) = x^n$, we get for the juxtapolynomial $\tilde{p}(x) \in \mathcal{P}_{n-1}$ of $x^n$ on $Q$ the condition

$$(1.10) \qquad \varepsilon(-1)^k \left(x_{i_k}^n - \tilde{p}(x_{i_k})\right) \geq 0 \quad \text{for} \quad k=1,2,\ldots,n+1.$$

In this case the polynomial $p^*(x) = x^n - \tilde{p}(x)$ is exactly a (real) extremal polynomial on $Q \subset R^1$, hence according to the previous result there are indices $1 \leq j_1 \leq j_2 \leq \ldots \leq j_n \leq m$ such that

$$(1.11) \qquad x_{j_k}^n - \tilde{p}(x_{j_k}) = 0 \quad \text{for} \quad k = 1,2,\ldots,n.$$

If all roots of $p^*$ are simple then all $x_{j_k}$ in (1.11) are distinct and in this case (1.11) implies (1.10).

## 2. Generalized juxtapolynomials when $\phi$ is a T-system.

Let $X$ be a totally ordered set with "$<$" as order relation (we use this also for real numbers, the meaning is clear from the context) and $Q \subset X$ be a finite set of the cardinality $m \geq n+1$. $\phi$ is called the Chebyshev/T/-system of order $n$ on $Q$ if

$$(2.1) \qquad \det \begin{bmatrix} \varphi_1(x_1), \varphi_1(x_2), \ldots, \varphi_1(x_n) \\ \vdots \quad \vdots \quad \vdots \\ \varphi_n(x_1), \varphi_n(x_2), \ldots, \varphi_n(x_n) \end{bmatrix}$$

is of a constant non-zero sign /say positive/ for all $x_1, x_2, \ldots, x_n \in Q$ such that $x_1 < x_2 < \ldots < x_n$ /see, e.g., [8],[9], [10],[11]/.

For a finite sequence of real numbers $\alpha_1, \alpha_2, \ldots, \alpha_m$, denote by $S^+(\alpha_1, \alpha_2, \ldots, \alpha_m)$ the maximal number of sign changes in the sequence achievable by appropriate assignment of signs $+1$ or

-1 to the zero entries /if any/ of the sequence /see [8],[9], [11]/. First, let us recall the lemma as follows.

*Lemma 1.* Let $\phi \subset F(X)$ be a T-system of order $n$ on $Q = \{x_1, x_2, \ldots, x_m\} \subset X$, $x_1 < x_2 < \ldots < x_m$, $m \geq n+1 \geq 2$. Then

$$S^+(\varphi(x_1), \varphi(x_2), \ldots, \varphi(x_m)) \leq n - 1 \tag{2.2}$$

for all $\varphi \in \phi$ and given any sequence $v_1, v_2, \ldots, v_m$ of numbers +1, -1 and 0 with $S^+(v_1, v_2, \ldots, v_m) \leq n - 1$, there is a $\varphi \in \phi$ such that

$$\begin{aligned} &\varphi(x_i) = 0 \quad \text{if} \quad v_i = 0, \\ &\varphi(x_i) > 0 \quad \text{if} \quad v_i = +1, \quad i=1,2,\ldots,m \\ &\varphi(x_i) < 0 \quad \text{if} \quad v_i = -1. \end{aligned} \tag{2.3}$$

□

*Proof:* The relation (2.2) due to Gantmacher and Krein [8] is a standard statement concerning Chebyshev systems /see also [9], [10], [11]/. The second part of the lemma /(2.3)/ is in fact an existence result for the following system of linear inequalities:

Given $a(i) = (\varphi_1(x_i), \varphi_2(x_i), \ldots, \varphi_n(x_i)) \in R^n$, $i=1,2,\ldots,m$, find $y \in R^n$ such that

$$\begin{aligned} &\langle a(i), y\rangle = 0 \quad \text{if} \quad v_i = 0, \\ &\langle a(i), y\rangle > 0 \quad \text{if} \quad v_i > 0, \quad i=1,2,\ldots m, \\ &\langle a(i)\ y\rangle < 0 \quad \text{if} \quad v_i < 0, \end{aligned} \tag{2.4}$$

/ $\langle . , . \rangle$ means the scalar product in $R^n$/,

For the proof that under the assumptions of the lemma the system (2.4) is solvable we refer to [12]/ see also [13]/. ■

Now, we have the following

*Theorem 1.* Let $\phi$ and $Q$ fulfil the assumptions of Lemma 1. The function $\varphi \in \phi$ is a g-juxtapolynomial of $f \in F(X) \setminus \phi$ on $Q$ if and only if

$$S^+(f(x_1)-\varphi(x_1),\ f(x_2)-\varphi(x_2),\ldots,\ f(x_m)-\varphi(x_m)) \geq n. \tag{2.5}$$

*Proof:* According to the definition of a g-juxtapolynomial it is enough to prove the theorem for $\varphi = \vartheta$. Let $\vartheta$ be the g-juxtapolynomial of $f$ on $Q$ and assume that

$$S^+(f(x_1),\ f(x_2),\ldots,\ f(x_m)) \leq n - 1. \tag{2.6}$$

By the lemma, (2.6) implies that there is $\varphi \in \phi$ such that

$$\begin{aligned} &\varphi(x_i) = 0 \quad \text{if} \quad f(x_i) = 0 \\ &\varphi(x_i) > 0 \quad \text{if} \quad f(x_i) > 0, \qquad i=1,2,\ldots,m, \\ &\varphi(x_i) < 0 \quad \text{if} \quad f(x_i) < 0. \end{aligned} \tag{2.7}$$

But (2.7) means that $\vartheta$ cannot be a g-juxtapolynomial of $f$ on $Q$.

Assume now that $\vartheta$ is not the g-juxtapolynomial of $f$ on $Q$. This means that there is $\varphi \in \phi$ such that (2.7) holds. In this case clearly the $S^+$ of the sequence $f(x_i)$ is equal to the $S^+$ of the sequence $\varphi(x_i)$. Using (2.2) in the lemma this implies that (2.6) holds. ■

## 3. Generalized juxtapolynomials when $\phi$ is a wT-system

Again, let $X$ be a totally ordered set and $Q$ be a finite subset of $X$ of the cardinality $m \geq n + 1$.

$\phi$ is called the weak Chebyshev /wT/-system of order $n$ on $Q$ if

$$(3.1)\qquad \varepsilon \cdot \det \begin{bmatrix} \varphi_1(x_1), & \varphi_1(x_2), \ldots, & \varphi_1(x_n) \\ & & \\ \varphi_n(x_1), & \varphi_n(x_2), \ldots, & \varphi_n(x_n) \end{bmatrix} \geqq 0$$

for some $|\varepsilon| = 1$ and for all $x_1, x_2, \ldots, x_n \in Q$, $x_1 < x_2 < \ldots < x_n$, and the determinant in (3.1) is zero for some $x_1, \ldots, x_n \in Q$, and non-zero for some $x_1, \ldots, x_n \in Q$

It turned out that *Theorem 1* does not hold for wT-systems. However, if we introduce instead of g-juxtapolynomials the concept of strict g-juxtapolynomials, then a similar result holds.

*Definition 3.* Using the notations of *Def.1*, $\vartheta$ is called the strict generalized /sg/-juxtapolynomial of $f$ on $Q$ if there is no $\varphi \in \Phi$ such that

$$(3.2)\qquad \begin{aligned} &\varphi(x) = 0 \quad \text{if} \quad x \in Q \quad \text{and} \quad f(x) = 0, \\ &\varphi(x) \geqq 0 \quad \text{if} \quad x \in Q \quad \text{and} \quad f(x) > 0, \\ &\varphi(x) \leqq 0 \quad \text{if} \quad x \in Q \quad \text{and} \quad f(x) < 0, \\ &\sum_{x \in Q} |\varphi(x)| > 0 \end{aligned}$$

□

*Definition 4.* We say that $\varphi \in \Phi$, $\varphi(x) \not\equiv 0$ on $Q$, is a strict generalized /sg/-juxtapolynomial of $f$ on $Q$ if $\vartheta$ is the sg-juxtapolynomial of $f - \varphi$ on $Q$. □

It is clear that (3.2) is more "restrictive" than (0.1). For example, a $\varphi \in \Phi$ which is nearest to $f$ in the distance function (0.3) is a g-juxtapolynomial but in general not an sg-juxtapolynomial.

For studying wT-systems we shall also need the following notion: Denote by $S^-(\alpha_1, \alpha_2, \ldots, \alpha_m)$ the minimal number of sign changes in the sequence $\alpha_1, \alpha_2, \ldots, \alpha_m$, $\alpha_i \in R^1$, achi-

evable by appropriate assignment of signs +1 and -1 to zero entries /if any/ of $\alpha_i$ /see, e.g., [8],[11]/. Clearly $S^-$ equals to the number of "proper" sign changes in the sequence when zeros are discharged from the sequence.
The first result in this section is the following

*Theorem 2.* Let $\Phi$ be a wT-system of order $n$ on $Q = \{x_i\}_{i=1}^{m} \subset X$, $x_1 < x_2 < \ldots < x_m$, $n+1 \leq m < \infty$. If $\varphi \in \Phi$ is an sg-juxtapolynomial of $f \in F(X)$ on $Q$, then

$$(3.3) \qquad S^+(f(x_1)-\varphi(x_1),\ f(x_2)-\varphi(x_2),\ldots,\ f(x_m)-\varphi(x_m)) \geq n. \quad \square$$

*Proof:* It is enough to prove the theorem for $\varphi = \vartheta$. Let $\vartheta$ be the sg-juxtapolynomial of $f$ on $Q$ and assume that

$$(3.4) \qquad S^+(f(x_1),\ldots,\ f(x_m)) \leq n-1.$$

For $\varrho > 0$ define

$$\varphi_j(x,\varrho) = \begin{cases} \sum_{i=1}^{m} e^{-(k-i)^2/\varrho^{j(x)}} \varphi_j(x_i) & \text{if} \quad x \notin Q \\ & \text{if} \quad x=x_k,\ k=1,\ldots,m \end{cases}$$

$$j = 1,2,\ldots,n \qquad x \in X.$$

It can be easily shown using the Couchy-Binet formula that if $\varphi_j$ is a wT-system on $Q$, then $\varphi_j(x,\varrho)$ is a T-system on $Q$ for all $\varrho > 0$. This implies, using *Lemma 1* and taking into account (3.4), that for all $\varrho > 0$ there is $y(\varrho) \in R^n$ such that $\| y(\varrho) \| = 1$ and

$$
\begin{aligned}
&\langle a(i,\varrho),y(\varrho)\rangle = 0 \quad \text{if} \quad f(x_i)=0\\
&\langle a(i,\varrho),y(\varrho)\rangle > 0 \quad \text{if} \quad f(x_i)>0\\
&\langle a(i,\varrho),y(\varrho)\rangle < 0 \quad \text{if} \quad f(x_i)<0
\end{aligned}
\tag{3.5}
$$

for all $i=1,\dots,m$, where $a(i,\varrho)=(\varphi_1(x_i,\varrho),\dots,\varphi_n(x_i,\varrho))\in R^n$. Taking a sequence $\varrho_1 < \varrho_2 <\dots< \varrho_r \dots$, $\varrho_r \to \infty$, the sequence of vectors $y(\varrho_r)\in R^n$, $r=1,2,\dots$, contains a convergent subsequence /because $y(\varrho)$ are in a compact set/. Denote the limit point of this subsequence by $y$. It is clear that

$$\varphi_j(x_i,\varrho_r) \to \varphi_j(x_i), \quad \text{for} \quad j=1,\dots,n, \quad i=1,\dots,m.$$

Hence taking limits in (3.5) we get: $\|y\| = 1$ and

$$
\begin{aligned}
&\langle a(i),y\rangle = 0 \quad \text{if} \quad f(x_i) = 0,\\
&\langle a(i),y\rangle \geq 0 \quad \text{if} \quad f(x_i)>0, \quad i=1,\dots,m\\
&\langle a(i),y\rangle \leq 0 \quad \text{if} \quad f(x_i) < 0,
\end{aligned}
\tag{3.6}
$$

where $a(i) = (\varphi_1(x_i), \varphi_2(x_i),\dots,\varphi_n(x_i))$ .
But this is a contradiction, because $\varphi(x) = \sum_{j=1}^{n} y_j \varphi_j(x)$ is not identically zero on $Q$ /being $\varphi_j$ linearly independent on $Q$/, hence (3.2) holds. ■

In the contrary to *Theorem 1*, (3.3) is in general only a necessary but not a sufficient condition for $\varphi$ to be an sg-juxtapolynomial of $f$. In [15] we give some "counterexamples" showing this fact.

The following theorem proves a sufficient condition for $\varphi$ to be a g-juxtapolynomial of $f$ on $Q$.

*Theorem 3.* Let $\phi$, Q be as in *Theorem 2* and let $f \in F(X) \setminus \phi$. If for $\varphi \in \phi$ the condition

$$(3.7) \qquad S^{-}(f(x_1)-\varphi(x_1),\ f(x_2)-\varphi(x_2),\ldots,\ f(x_m)-\varphi(x_m)) \geq n$$

holds, then $\varphi$ is a g-juxtapolynomial of f on Q. □

*Proof:* The proof is very simple and depends on the following well known fact concerning wT-systems: for any $\varphi \in \phi$

$$(3.8) \qquad S^{-}(\varphi(x_1),\varphi(x_2),\ldots,\varphi(x_m)) \leq n - 1,$$

/see, e.g., [8],[11]/.

It is enough to prove the theorem for $\varphi = \vartheta$. Assume that $\vartheta$ is not a g-juxtapolynomial of f on Q. This means that there is $\varphi \in \phi$ such that

$$\varphi(x_i) = 0 \quad \text{if} \quad f(x_i) = 0,$$

$$\varphi(x_i) > 0 \quad \text{if} \quad f(x_i) > 0,$$

$$\varphi(x_i) < 0 \quad \text{if} \quad f(x_i) < 0.$$

This implies $S^{-}(\varphi(x_1),\ldots,\varphi(x_m)) = S^{-}(f(x_1),\ldots,f(x_m))$ and using (3.8) we come to a contradiction with (3.7) /$\varphi = \vartheta$/. ■

The condition (3.7) is far from being a necessary condition for $\varphi$ to be a g-juxtapolynomial of f on Q, /see [15] for more details/.

## 4. Remarks, comparing the results with known ones

1. Let us recall the original definition of g-juxtapolynomials /see [4],[7]/:

*Definition 5.* $\varphi \in \Phi$ is a g-juxtapolynomial of $f$ on $Q$ if there is no $\psi \in \Phi$ such that

$$(4.1) \qquad |f(x)-\psi(x)| \begin{cases} = 0 & \text{if} \quad f(x) = \varphi(x) \\ < |f(x)-\varphi(x)| & \text{if} \quad f(x) \neq \varphi(x) \end{cases}$$

hold for all $x \in Q$. □

We can prove the following

*Assertion 1.* The *Definition 2* is equivalent to the *Definition 5.* □

*Proof:* Assume that there is $\xi \in \Phi$ such that

$$(4.2) \qquad \begin{aligned} \xi(x) &= 0 \quad \text{for} \quad x \in Q : f(x) = \varphi(x) , \\ \xi(x) &> 0 \quad \text{for} \quad x \in Q : f(x) > \varphi(x) , \\ \xi(x) &< 0 \quad \text{for} \quad x \in Q : f(x) < \varphi(x) . \end{aligned}$$

/I.e. $\varphi$ is not a g-juxtapolynomial according to *Def.2*/

Then for any $0 < \varepsilon < \min \left\{ \frac{f(x) - \varphi(x)}{\xi(x)} : \xi(x) \neq 0 \right\}$ we have for all $x \in Q$:

$$(4.3) \qquad |f(x)-\varphi(x) - \varepsilon \xi(x)| \begin{cases} = 0 & \text{if} \quad f(x) = \varphi(x) \\ < |f(x) - \varphi(x)| & \text{if} \quad f(x) \neq \varphi(x). \end{cases}$$

Denoting $\psi = \varphi + \varepsilon\xi$ this show that $\varphi$ is not a g-juxtapolynomial according to *Def. 5.*
Conversely, if there is $\psi \in \Phi$ such that (4.1) holds, then we can easily see that $\xi = \psi - \varphi$ fulfils (4.2).
The condition (0.1) of *Def.2* /Def.1/ seems to be more easily to check than the condition (4.1).
Motzkin and Walsh [4] also proved a characterization theorem for algebraic juxtapolynomials, see *Section 1.*
They introduced a following concept:
We say that a sequence of real numbers $\alpha_1, \alpha_2, \ldots, \alpha_m$, $m \geq n+1$, has at least $n$ weak sign changes if there are indiced $1 \leq i_1 < i_2 < \ldots < i_{n+1} \leq m$ and $|\varepsilon| = 1$ such that

$$\varepsilon(-1)^k \alpha_{i_k} \geq 0, \quad k = 1,2,\ldots,n+1. \tag{4.4}$$

Then their theorem is: $p \in P_{n-1}$ /algebraic polynomials of the degree at most $n-1$/ is a juxtapolynomial of $f \in F(R^1) \setminus P_{n-1}$ on $Q = \{x_1, \ldots, x_m\} \subset R^1$, $x_1 < x_2 < \ldots < x_m$, $m \geq n+1$, if and only if the sequence $f(x_1)-p(x_1)$, $f(x_2)-p(x_2)$, ..., $f(x_m)-p(x_m)$ has at least $n$ weak sign changes.
Rice [7] generalized this result to the case when $X = [0,1]$ and $\Phi \subset C([0,1])$ is a continuous T-system of order $n$ /he used for $\varphi \in \Phi$, for which the sequence $f(x_i)-\varphi(x_i)$ has at least $n$ weak sign changes, the term "weakly interpolating g-polynomial"/.
It can be proved that our and their conditions are equivalent, namely the following assertion holds.

*Assertion 2.* The sequence of real numbers $\alpha_1, \alpha_2, \ldots, \alpha_m$, $m \geq n+1$, has at least $n$ weak sign changes if and only if

$$S^+(\alpha_1, \alpha_2, \ldots, \alpha_m) \geq n. \tag{4.5}$$

For the proof of this assertion /which is not complicated/ we refer to a forthcoming paper [16] where many other statements

regarding the combinatorial properties of finite sequences of signs are also proved.
Again, the condition (4.5) seems to be checked more easily than the existence of indices $\{i_k\}_{k=1}^{n+1}$ satisfying (4.4): while for the computing of $S^+(\alpha_1)$ at most $m-1$ comparisons are needed, this number is $n\binom{m}{n+1}$ in the case of (4.4). Of course, these are very "rough" estimates, both can be rapidly decreased using some finer combinatorial investigations, see [16]. This is especially true for the checking of (4.5).

2. As regards the results of Section 3, both the concepts used /e.g. strict g-juxtapolynomials/ and the results proved seem to be new ones.
Let us formulate the notion of sg-juxtapolynomials in a form similar to that in Definition 5.

*Definition 6.* $\varphi \in \Phi$ is a strict g-juxtapolynomial of $f$ on $Q$ if there is no $\psi \in \Phi$ such that

$$|f(x)-\psi(x)| \begin{cases} = 0 & \text{if } f(x)=\varphi(x), \\ \leq |f(x)-\varphi(x)| & \text{if } f(x)\neq\varphi(x), \end{cases} \tag{4.6}$$

hold for all $x \in Q$ and $|f(y)-\psi(y)|<|f(y)-\varphi(y)|$ for at least one $y \in Q$ such that $f(y)\neq\varphi(y)$. □
Similarly to Assertion 1 we can prove the following
*Assertion 3.* The Definition 4 /Sec.3/ is equivalent to the Definition 6. □

*Proof:* Analogous to that of Assertion 1. ■

The results of Sec.3 have to be considered as first modest ones. The deeper study of wT-systems will probably yield sharper results. The importance of wT-systems is increased by the fact, that spline functions constitute such a system.

*REFERENCES*

[1] L. Fejér, Über die Lage der Nullstellen von Polynomen die aus Minimumfolgerungen gewisser Art entspringen, Math. Ann., 85, 42-48 /1922/.

[2] M. Fekete, J. von Neumann, Über die Lage der Nullstellen gewisser Minimumpolynome, Jahresb. Deutsch. Math. Ver., 31, 125-138 /1922/.

[3] M. Fekete, On the structure of polynomials of least deviation, Bull. Res. Council Israle, 5A, 11-19 /1955/.

[4] T.S. Motzkin, J.L. Walsh, Polynomials of best approximation on a real finite point set I, Trans. Amer. Math. Soc., 91, No.2, 231-245 /1959/.

[5] O. Shisha, Best approximation on some finite sets, J. of Math. Anal. Appl., 21, 347-355 /1968/.

[6] I. Marusciac, On linear juxtaoperators, in: Z. Cieselski, J. Musielak, eds., "Approximation Theory", PWN, Warszawa, 1975.

[7] J. Rice, "Approximation of Functions. Vol II: Non-linear and Multivariate Theory", Addison-Wesley, Reading, Mass., 1969.

[8] В. Гантмахер, М.Г. Крейн, "Осцилационные матрицы и малые колебания механических систем", ГТТИ, Москва, 1950.

[9] М.Г. Крейн, А.А. Нудельман, "Проблема моментов Маркова и экстремальные задачи", Наука, Москва, 1973.

[10] S. Karlin. W.J. Studden, "Tschebysheff-Systems: With Applications to Anal. and Stat.", Interscience, N.Y.,1966.

[11] S. Karlin, "Total Positivity", Stanford Univ. Press, 1968.

[12] B. Uhrin, "Theorem of Helly and the Existence of Generalized Polynomials with Prescribed Values and Signs", Seminar Notes, Mathematics No.3, Hungarian Committee for Systems Analysis, Budapest, 1975.

[13] A. Tihanyi, B. Uhrin, About the generalized polynomials with prescribed signs on discrete sets, Ann. Univ. Sci. Budapest, Sec. Math., 19, 165-169 /1976/.

[14] B. Uhrin, A characterization of finite Chebyshev sequences in $R^n$, Lin. Algebra Appl., 18, 59-74 /1977/.

[15] B. Uhrin, A characterization of generalized juxtapolynomials on finite point sets, Coll. Math. Soc. J. Bolyai, to appear.

[16] B. Uhrin, Some combinatorial results concerning finite sequences of signs, in preparation

## ЕДИНЫЙ ПОДХОД К ТЕОРИИ ПРИБЛИЖЕННИЙ НА КОНЕЧНЫХ МНОЖЕСТВАХ

Б. Ухрин

### Резюме

В статье вводится понятие "обобщенного юкста-многочлена" как элемента подпространства функций, который является "самым близким" к данной функции и дается их характеристика в случае Чебышевских подпространств. Несколько результатов касающихся слабых Чебышевских подпространств тоже доказано в статье.

MTA SZTAKI Tanulmányok 147/1983 101-109

# A RECURSION THEORY CHARACTERIZATION OF INDUCTIVE INFERENCE WITH ADDITIONAL INFORMATIONAL CLASSES

O.I. Botusharov

Institute of Mathematics with Computing Centre
Bulgarian Academy of Sciences

The mathematical theory known generally as inductive inference (or algorithmic identification) is based on the black-box principle. According to it the knowledge of some black-box-informations on a given system creates the possibility to derive certain statements, describing its strucutre. In the present paper we use the following modification of this principle: The task to synthesize a program for a certain general recursive function $f$ is given to a "program-constructor". No description of $f$ in a language, that he can understand, is available. The "program-constructor" works in steps and in any of them a program-hypothesis has to be built up, using only some inputs 1,2,...,n and the corresponding outputs $f(1),f(2),\ldots,f(n)$. The increase in the number of steps must leed to a stabilization of the program-hypothesis on a real program of $f$ ($f$ is being identified in the limit). This problem was discussed for the first time by GOLD in [5].

There are situations in which the "program-constructor" is supplied with some extra knowledge about $f$: e.g. belongs to a certain class of functions, a program for $f$ is known to have a definite complexity etc. The whole process is then refered to as inductive inference with additional information. The presence of additional information may substantially increase the power of the "program-constructor" or under certain conditions have no impact on it. Naturally there are many types of additional information. In the following we shall consider the one introduced by FREIWALD/WIEHAGEN in [4].

We can formalize the above described process of inductive inference with additional information as follows: the "program-constructor" is a partial recursive function, that is also called identification strategy; a program for $f$ is a number of $f$ in a fixed Gödel numbering of all partial recursive function/see [8] /; any upper bound of the minimal number of $f$ in this Gödel numbering is considered additional information. This is the usual formalization applied by BARZDIN [2], BLUM/BLUM [1] and WIEHAGEN |9|.

An identification strategy is able to synthesize programs for a whole class of general recursive functions. Such strategies can in addition have some practically important properties: consistency of the hypotheses to the function to be identified, finite number of steps leeding to the stabilization etc. In [7], [8] and [11] the power of identification strategies, characterized by the volume of the corresponding families of identified functional classes, is investigated. In [4] this is done in relation to additional information. The families of identified functional classes are refered to as identification types. These have been studied using means, provided by the theory of inductive inference itself. However it is important to consider the identification types from a different point of view with the purpose of obtaining deeper results. We can achieve this by means of an inductive inference-free characterization of the identification types. Characterization results were obtained by WIEHAGEN/LIEPE [11] - using means from the theory of com_ plexity, JUNG [6] - using the apparatus of topology and WIEHAGEN/JUNG [10] - exploiting only the possibilities and means, given by the classical recursion theory. Other characterization problems have been solved by BLUM/BLUM [1] and FREIVALD [3]. All kinds of characterization consider the usual identification types. The following paper presents recursion theory characterizations of the identification types which require additional information. A similar topological or theory of complexity-characterization of these remains an open problem.

Next we turn to the precise definitions and formulation of results.

Let $Nz$ denote the set of all natural numbers, $P^n$ and $R^n$ denote the class of all partial recursive respectively general recursive function of $n \in Nz$ variables. For $n=1$ we omit the upper index. Further let $\varphi \in P^2$ denote a fixed Gödel numbering of $P$. For $f \in P$ the minimal number in $\varphi$ is denoted by $min_\varphi f$. If $f(x)$ is defined for all $x \leq n$, then let $f[n]$ denote a CANTOR-number of $(f(0),\ldots,f(n))$. If $L$ and $M$ are sets, then $P(L)$ denotes the power set of $L$ and $L \subset M$ denotes the proper inclusion of $L$ in $M$. As usual $\mu\ldots[\ldots]$ denotes the $\mu$-operator and $\lambda$-notation is omitted for the sake of convencience.

We say that the sequence $(x_n)_{n \in Nz}$ of natural numbers is convergent to a number $x \in Nz$ iff $n_o \in Nz$ exists such that $x_n = x$ for all $n \geq n_o$, and denote it by $x = \lim_n x_n$. In the following we shall also use a modification of this definition: a sequence $(x_n)_{n \in Nz}$ of natural numbers is said to be finitely convergent to a number $x$ iff it is convergent to $x$ and for any $n_o$ such that $x_{n_o} = x_{n_o+1}$ it already follows that $x_n = x_{n_o}$ for any $n \geq n_o$. We denote this by $x = \lim_n{}_e\, x_n$. The following identification type definition was first given by GOLD in [5].

DEFINITION 1: Let $U \subseteq R$. $U$ is <u>identifiable in the limit</u> iff there is a strategy $F \in P$ such that for any function $f \in U$ holds:

1) $F(f[n])$ is defined for any $n \in Nz$,

2) $a = \lim_{Df} F(f[n])$ exists

and

3) $\varphi_a = f$.

We denote the fact that $U$ is identifiable in the limit by $U \in LIM$ and say that $U$ is of identification type $LIM$.

The next identification types were first considered by WIEHAGEN [9] and WIEHAGEN/JUNG [10].

DEFINITION 2: Let $U \subseteq R$. $U$ is identifiable in the limit <u>by a consistent strategy</u> $(U \in CONS)$ iff there is a strategy $F \in P$ such that 1), 2) and 3) from DEFINITION 1
and
4) For any $n \in Nz$ and any $x \leq n$ is $\varphi_{F(f[n])}(x) = f(x)$.

DEFINITION 3: Let $U \subseteq R$. $U$ is identifiable in the limit <u>by an $R$-consistent strategy</u> $(U \in R\text{-}CONS)$ iff there is a strategy $F \in R$ such that:
1) For any $f \in U$ holds:
   a) $a = \underset{Df}{lim}\ F(f[n])$ exists
and
   b) $\varphi_a = f$
2) For any $f \in R$ , any $n \in Nz$ and any $x \leq n$ is $\varphi_{F(f[n])}(x) = f(x)$.

DEFINITION 4: Let $U \subseteq R$. $U$ is <u>finitely identifiable</u> $(U \in FIN)$ iff there is a strategy $F \in P$ such that for any function $f \in U$ holds:
1) $F(f[n])$ is defined for any $n \in Nz$,
2) $a = \underset{Df}{lim}_{e}\ F(f[n])$ exists
and
3) $\varphi_a = f$.

The last three definitions pay tribute to the practically important properties of the identification strategies that were previously mentioned.

The following statement comprises results obtained by WIEHAGEN in [9] and LINDER in [7].

THEOREM 1: $\begin{matrix} FIN \\ \supset \\ R\text{-}CONS \end{matrix} \subset CONS \subset LIM \subset P(R)$

We turn now to identification types in which additonal information is used.

DEFINITION 5: Let $U \subseteq R$. $U$ is <u>identifiable in the limit with additional information</u> $(U \in LIM^{+})$ iff there is a strategy $S \in P^{2}$ such that for any function $f \in U$ and any $b \geq min_{\varphi}\ f$ holds:

1) $S(b,f[n])$ is defined for any $n \in Nz$,

2) $a = \lim_{n} S(b,f[n])$ exists
$\quad\quad Df$

and

3) $\varphi_a = f$.

In a similar way one can define $CONS^+$, $R\text{-}CONS^+$ and $FIN^+$. FREIVALD/WIEHAGEN [4] have studied the changes in the power of identification strategies, which occur due to the use of additional information. They proved following theorems.

THEOREM 2: $CONS^+ = LIM^+ = P(R)$

THEOREM 3: $R\text{-}CONS = R\text{-}CONS^+$

THEOREM 4: $FIN \subset FIN^+ \subset P(R)$

It is obvious that additional information has a different impact on the various identification types. This makes an adequate characterization even more necessary. WIEHAGEN/JUNG [10] give a recursion theory characterization of all identification types from THEOREM 1 with the exception of $R\text{-}CONS$. A simple characterization of $CONS^+$ and $LIM^+$ is given by THEOREM 2. The following results give a recursion theory characterization of $R\text{-}CONS^+$ and $FIN^+$. Since $R\text{-}CONS = R\text{-}CONS^+$ it is adequate to consider a characterization of $R\text{-}CONS$ only.

THEOREM 5: Let $U \subseteq R$. $U \in R\text{-}CONS$ iff there are general recursive functions $h,\nu \in R$, such that following conditions hold:

1) a) For any function $f \in U$ exists a number $i$ such that $\varphi_{h(i)} = f$,
   b) For any $n \in Nz$ and arbitrary $y_0,\ldots,y_n \in Nz$ there is a number $i$ such that $\varphi_{h(i)}[n] = (y_0,\ldots,y_n)$;

2) For any $n \in Nz$ and arbitrary $y_0,\ldots,y_n \in Nz$ holds:
$\nu(y_0,\ldots,y_n) = \mu i[\varphi_{h(i)}[n] = (y_0,\ldots,y_n)]$.

PROOF: Necessity:

Let $F$ be a strategy identifying $U \subseteq R$ in $R\text{-}CONS$-sense. Further

let $a \in R$ enumarete without repetition the set

$\{(j,n)/\forall x \leq n: \varphi_j(x), \varphi_{F(\varphi_j[n])}(x)$ is defined $\wedge \varphi_{F(\varphi_j[n])}(x)=\varphi_j(x)\}$

We select now $h \in R$ such that for any $i \in Nz$, $c(i)=(j,n)$ holds:

$$\varphi_{h(i)}(x)=\begin{cases}\varphi_{F(\varphi_j[n])}(x), & \text{if } x \leq n\\ \varphi_{F(\varphi_j[n])}(x), & \text{if } x>n \text{ and for all } x' \text{ such that } n<x' \leq x\\ & \varphi_j(x') \text{ and } \varphi_{F(\varphi_j[n])}(x') \text{ are defined and}\\ & F(\varphi_j[x'])=F(\varphi_j[n]) \text{ holds}\\ \text{undefined otherwise}\end{cases}$$

Then condition 1) holds.

For $g(i) \underset{Df}{=} n$ *with* $c(i)=(j,n)$ the function $\nu \in R$ is defined as follows:

$\nu(y_0,\ldots,y_n) \underset{Df}{=}$ "Search one $i_o$ such that $\varphi_{h(i_o)}(x)=y_x$ for all $x \leq n$.

Let $i^*$ be the smallest $i \leq i_o$, such that either it is $g(i) \geq n$ (then $\varphi_{h(i)}(x)$ is defined for all $x \leq n$) and $\varphi_{h(i)}(x)=y_x$ for all $x \leq n$ or it is $g(i)<n$ and $\varphi_{h(i)}(x)=y_x$ for all $x \leq g(x)$ and $F(y_0,\ldots,y_x)=F(y_0,\ldots,y_{g(i)})$ for all $x$ such that $g(i)<x \leq n$.

Then define $\nu(y_0,\ldots,y_n)=i^*$."

Because of condition 1) for any function $f \in U$ and for any $n \in Nz$ there is always a number $i_o$ such that $\varphi_{h(i_o)}[n]=f[n]$. Condition 2) is then secured by the definition of $h$.

Sufficiency:

A strategy $F \in R$ is defined as follows:

$F(y_0,\ldots,y_n) \underset{Df}{=} h(\nu(y_0,\ldots,y_n))$.

Obviously $F$ identifies the class $U$ in the $R$-$CONS$-sense. This completes the proof.

THEOREM 6: Let $U \subseteq R$. $U \in FIN^+$ iff there are recursive functions $h \in R$, $p \in P^2$ such that following conditions hold:

1) For any function $f \in U$ exists a number $i$ such that $\varphi_{h(i)} = f$;

2) For any function $f \in U$ and any numbers $i, b \in Nz$ holds:

   a) If $\varphi_{h(i)}$ $f$ and $b \geq min\ f$ holds, then $p(i,b)$ is defined,

   b) If $p(i,b)$ is defined, then from $\varphi_{h(i)} \underset{p(i,b)}{=} f$ follows already $\varphi_{h(i)} = f$. ($\underset{x}{=}$ means "equal up to the argument value $x$")

PROOF: Necessity:
Let $S \in P^2$ be a strategy identifying $U \subseteq R$ in the $FIN^+$-sense. We consider the set

$$\{(j,n(b)) / \forall x \leq n+1 : \varphi_j(x), S(b, \varphi_j[x]) \text{ is defined} \wedge n(b) = \mu x [S(b, \varphi_j[x]) = S(b, \varphi_j[x+1])]\}$$

It can be enumerated without repetition by a general recursive function $a$. Then for $i \in Nz$ and $a(i) = (j, n(b))$ we define:
$h(i) \underset{Df}{=} j$ and $p(i,b) \underset{Df}{=} n(b)$.
It is quite obvious that for these functions hold conditions 1) and 2).
Sufficiency:
We construct a strategy $S \in P^2$ as follows:
$S(b,0) \underset{Df}{=} 0$ for any $b \in Nz$.
Let now $n > 0$, $1 = (y_0, \ldots, y_n)$, $1' = (y_0, \ldots, y_{n-1})$ and $b \in Nz$ be arbitrary. Then $S(b,1)$ is defined as follows:

$S(b,1) \underset{Df}{=}$ "If $S(b,1')$ is defined and $S(b,1') \neq n-1$, then it is $S(b,1) = S(b,1')$. IF $S(b,1') = n-1$, then check whether there is one number $i \leq n$ such that in a maximum of $n$ steps it can be decided whether $p(i,b)$ is defined, $p(i,b) \leq n$ holds and whether for any $x \leq p(i,b)$ the value $\varphi_{h(i)}(x)$ is computable in a maximum of $n$ tacts, and for any $x \leq p(i,b)$ holds $\varphi_{h(i)}(x) = y_x$.
If no, then $S(b,1) = n$.
If yes, then let $i^*$ be the smallest such $i$.
Define $S(b,y) = h(i^*)$."

It is easily seen that $S$ identifies the class $U$ in the $FIN^+$-sense. This completes the proof of our theorem.

REFERENCES

[ 1] BLUM,L., M.BLUM, Toward a Mathematical Theory of Inductive Inference. Infomr. and Control 88 (1975), 125-155.

[ 2] BARZDIN: Барздинь Я.М., Индуктивный вывод автоматов, функций и программ.
Internat.Math.Congress, Vancouver, 1974.

[ 3] FREIVALD,R.V., Minimal Gödel Numbers and Their Identification in the Limit. In: MFCS 1975; (Ed.J.Bečvar); Lecture Notes in Computer Science, 32; Berlin - Heidelberg - New York 1975; 219-225.

[ 4] FREIVALD,R.V., R. WIEHAGEN, Inductive Inference with Additional Information. EIK 15 (1979) 4, 179-185.

[ 5] GOLD, E.M., Limiting Recursion. J.Symb.Log. 30 (1965), 28-48.

[ 6] JUNG,H., Topological Characterizations in the Theory of Inductive Inference (to be published).

[ 7] LINDNER,R., Algorithmsche Erkennung. Dissertation B, Friedrich-Schiller-Universität Jena, 1972.

[ 8] ROGERS,H. Jr.,.Theory of Recursive Functions and Effective Computability.McGraw Hilll, New York, 1967.

[ 9] WIEHAGEN,R., Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. EIK 12 (1976) 1/2, 93-99.

[10] WIEHAGEN,R.,H.JUNG, Rekursionstheoretische Charakterisierung von erkennbaren Klassen rekursiver Funktionen. EIK 7/8, 385-397.

[11] WIEHAGEN,R., W.LIEPE, Charakteristische Eigenschaften von erkennbaren Klassen rekursiver Funktionen. EIK 12 (1976) 8/9 421-438.

# РЕКУРСИВНО-ТЕОРИТИЧЕСКАЯ ХАРАКТЕРИЗАЦИЯ КЛАССОВ ИНДУКТИВНОГО ВЫВОДА С ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИЕЙ

О.И. Ботушаров

Резюме

В работе рассматривается общая постановка теории индуктивного вывода. Вводится понятие дополнительной информации. Приводятся результаты, которые дают рекурсивно-теоритическую характеризацию классов индуктивного вывода с дополнительной информацией.

MTA SZTAKI Tanulmányok 147/1983 111-115

# A PAPER ON THE STRUCTURAL CHARACTERISTICS OF THE GRAPHS OF THE FUNCTIONS OF ONE CLASS

A. Aslanski
Sofia University in Blogoevgrad

This paper is on the structural characteristics, in a position to the separable pairs, of the graphs of functions $f(x_1,\dots,x_n)$ of the order of $n \geq 7$, for which exist two variables $x_i$ and $x_j$, such that the multitude $\{x_i, x_j\}$ is separable for $f$ and the subgraph of the graph $f$, having the elements of the multitude $\{x_1,\dots,x_n\}\setminus\{x_i, x_j\}$ for its apexes, looks like an elementary loop.

The terminology and symbols used, are from [1 - 16].

Theorem 1. If for the function $f(x_1,\dots, x_n)$ of the order of $n \geq 7$, the multitude $\{x_i, x_j\}$ is separable and the subgraph of the graph, of the multitude $\{x_1,\dots,x_n\}\setminus\{x_i, x_j\}$ looks like an elementary loop, it is true that at least one of the variables $x_i, x_j$ is of the order of $n-1$ for $f$.

Proof. Under the existing circumstances for the function $f(x_1,\dots,x_n)$ of the order of $n \geq 7$, let accept that $\{x_{n-1}, x_n\} \in S_f$ and the subgraph of $f$ with apexes $x_1,\dots, x_{n-2}$ looks like an elementary loop of the kind

$$\{x_1, x_2\}, \{x_2, x_3\},\dots, \{x_{n-3}, x_{n-2}\}, \{x_{n-2}, x_1\} \in S_f .$$

We'll prove that if some of the variables $x_{n-1}, x_n$ is of an order no bigger than $n-2$ for the function $f$, the other of these variables is of an order $n-1$ for $f$.

Let accept that $x_{n-1}$, is of an order not bigger than $n-2$ for $f$ . As $\{x_{n-1}, x_n\} \in S_f$, $x_{n-1}$ doesn't form a separable pair for $f$ at least with one of the variables $x_1, \ldots, x_{n-2}$.

Without restricting the community of investigation, let accept that $\{x_{n-1}, x_1\} \notin S_f$.

Let take up whichever three of variables $x_1, x_i, x_{i+1}$, $i \in \{3, \ldots, n-4\}$.

Only one separable pair exists for $f$ , which consists of variables of the multitude $\{x_1, x_i, x_{i+1}\}$ . Then, according to theorem 2 of [6] , there will be a variable which forms separable pairs for $f$ with everyone of the variables $x_1, x_i, x_{i+1}$ . It can be only the variable $x_n$ .

In this way we prove that $x_n$ forms separable pairs for $f$ with everyone of the variables $x_j$, $j \in \{1, \ldots, n-3\} \setminus \{2\}$.

If we use theorem 2 of [6] for the triads $x_1, x_2, x_4$ and $x_1, x_{n-2}, x_3$ , it can be proved that $x_n$ forms separable pairs for $f$ with the variables $x_2$ and $x_{n-2}$ too.

We proved, that $x_n$ is of the order of $n-1$ for $f$ .

Thus the theorem is proved.

It can be shown that there are functions which satisfy the condition of theorem 1.

Example. Let take up the function from the four-place logics

$$f = x_1 x_2^0 + (x_3 x_4^0 + x_4^1 x_5) + (x_5 x_6^0 + x_6^1 x_7) + x_3 x_7 x_2^3 \pmod 4,$$

where $$x_i^\alpha = \begin{cases} 1 & \text{when } x_i = \alpha \\ 0 & \text{when } x_i \neq \alpha \end{cases}$$

The subgraph of $f$ with apexes $x_3, x_4, x_5, x_6, x_7$ looks like an elementary loop and $\{x_1, x_2\} \in S_f$. And the variable $x_2$ is of

the fifth order for $f$ , regard to the separable pairs.

We'll mark that the statement in the theorem is not true for $n = 6$ . For example, let $f(x_1, \ldots, x_6)$ has for separables only the next two-element multitudes

$$\{x_1, x_4\}, \{x_3, x_4\}, \{x_2, x_6\}, \{x_5, x_6\}$$

$$\{x_1, x_i\}, \quad i = 2, 5, 6 \quad \text{and} \quad \{x_3, x_j\}, \quad j = 2, 5, 6$$

Such functions exist (e.g. the function $f_{12}$ of [16] ).

The multitude $\{x_5, x_6\}$ is separable for $f$ and the subgraph $f$ with apexes $x_1, x_2, x_3, x_4$ looks like an elementary loop. And yet, neither $x_5$ nor $x_6$ is of the fifth order for $f$ .

Theorem 1.is not true too when $n = 5$ . E.g. Let $f(x_1, \ldots, x_5)$ has for separables only two-element multitudes,

$$\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \{x_4, x_1\}, \{x_5, x_i\}, i = 1, 2, 3, 4$$

The pairs $\{x_1, x_2\}$ is separable for $f(x_1, \ldots, x_5)$ and the subgraph for $f$ with apexes $x_3, x_4, x_5$ looks like an elementary loop. And yet, neither $x_1$ nor $x_2$ is of the fourth order for $f(x_1, \ldots, x_5)$

LITERATURE USED:

1. Яблонский С.В. Функциональные построения в к-значной логике. Труды математического института им. В.А.Стеклова, т.51, 1958, 5 - 142.

15. Chimev K. N., M. Aslanski. Structural characteristics of the functions of one class. A paper, read on the jubilee scientific session of the Lesotehnicheski University, devoted to the 1300-anniversary of the foudation of Bulgaria, S. 1981.
16. Chimev K. N. Separable pairs of the functions of six arguments. Applied Mathematics, XII, book 2, 143 - 156.

## О СТРУКТУРНЫХ СВОЙСТВАХ ГРАФОВ ФУНКЦИЙ ОДНОГО КЛАССА

М. Асланski

Резюме

В работе исследуются структурные свойства (в отношении выделимых пар) графов функций $f(x_1, \ldots, x_n)$ порядка $n \geq 7$, для которых существуют две таких переменных $x_i$ и $x_j$, что множество $\{x_i, x_j\}$ выделимо для $f$ и подграф графа $f$ (с вершинами элементами множества $\{x_1, \ldots, x_n\} \setminus \{x_i, x_j\}$) имеет вид элементарной замкнутой цепи.

Используется терминология и обозначения от [1 - 16].

# DATA BASES AND MATHEMATICAL LINGUISTICS

# БАЗА ДАННЫХ И МАТЕМАТИЧЕСКАЯ ЛИНГВИСТИКА

MTA SZTAKI Tanulmányok 147/1983 119-124

# ИНТЕГРИРОВАНИЕ ДАННЫХ ГЕТЕРОГЕННЫХ СИСТЕМ

Атанас Терзиев
Институт математики с ВЦ, Болгарская Академия наук

Развитие теории моделей данных и систем управления баз данных привели к созданию и реальной эксплуатации множества разнородных систем. Широкое применение вычислительной техники и сети ЭВМ в человеческой деятельности предъявили новые требования к средствам обработки информации, в том числе и интегрирование обработок данных гетерогенных систем.

Важная проблема при интегрировании данных - это преобразование представлений данных. Уже несколько лет специалисты в области информатики занимаются его решением. Основное средство разрешения проблемы они видят в программных трансформациях, т.е. ретрансляция проблемных программ и преобразование процедурных операторов языка манипулирования данных (ЯМД) в непроцедурные реляционные вычисления и наоборот.

Хаузел /1-3/ предлагает систему, в которой первичные операторы программы преобразуются на основе "декомпиляции" в абстрактные спецификации, выражающиеся в терминах проблемной модели и баз данных. Потом ретранслируются в ЯМД этой системы. Однако, очень трудно формулировать механизм для выражения программ в терминах конкретных проблемных данных.

Другой подход разработывается в университете в Флориде /4-5/. Предлагается семантическая модель данных для характеризации пути, по которому данные будут использоваться прикладными программами. Это осуществляется на основе "графы пути доступа". Конструируется и "графа вопросов". Программы ретранслируются, отыскивая последовательность от инструкции в конкретного ЯМД, которая соответствует шаблонам доступа этих графов. К сожалению, авторы не успели показать актуальный алгоритм

реализующий выше отмеченные преобразования.

Существенный вклад в развитии этого направления внесли Вонг и Катц /6,7,11/. Они создали успешный алгоритм преобразования последовательности из процедурных операторов в непроцедурные спецификации, интегрируя процесс преобразования с информацией из спецификаций базы данных типа КОДАСИЛа.Модель пути доступа, использованная при проектировании физической БД, используется для определения семантического доступа внутри программы.

Дополнительную литературу, связанную с данной темой можно увидеть в /8-10/.

Наш подход несколько отличается от выше упомянутых. Он ориентирован главным образом на преобразование моделей данных к псевдо-реляционному табличному представлению. Метод заключается в создании виртуального информационного объекта (ВИО), содержащего описание того, какие поля из записей, достигаемых через пути доступа, должны быть включены и как они должны быть представлены. На ВИО дается ссылка в описании другого ВИО и т. д. Данные, соответствующие экземплярам конкретного ВИО, образуют виртуальную таблицу (ВТ). Для одного реального объекта можно создать несколько ВИО, в зависимости от цели обработок. Таким способом описание модели данных редуцируется в описание несколько ВИО и механизмов пути доступа.

Обработка данных редуцируется в обработку виртуальных таблиц, которая осуществляется с помощью специализированного внутренного ЯМД. Используя информации ВИО, операторы ЯМД конкретной СУБД трансформируют инструкции в ВЯМД и наоборот. Эти трансформации надо осуществлять "коммуникационными модулями", учитывающими специфику конкретной модели и ЯМД. модули выполняют и преобразование информации из записей физических баз данных в "записси" виртуальных таблиц и наоборот.

Таким способом получается стандартный интерфейс - внутренний ЯМД и виртуальные таблицы. Преобразование данных из од-

ной модели в другую означает преобразование в табличном представлении, а отсюда -в другую модель. Надо отметить, что связывая ВИО и ВТ определенным способом можно получить "вторичную" модель данных, т.е. установить новые семантические и физические связи между данными в уже существующих базах данных. Это означает получить новые "пораженные" виртуальные таблицы.

На основе высказанных выше идей , в Математическом институте БАН началась разработка система ДАКОМС. Это комплекс программного обеспечения, предназначенный для решения информационных задач. Она является СУБД, а одновременно и средством управления существующих СУБД. Система организована йерархически, предоставляет все свои уровни (в том числе и достаточно низкие) для возможного использования. Наличие описаний внешних и внутренних интерфейсов позволяет модифицировать систему, использовать ее части автономно.

Интерфейс с физическими БД осуществляется коммуникационными модулями. Кроме этого, есть собственный метод доступа и модули для обработки физических файлов, поддерживаемых операционной системой. Придуман еще один тип модуля - "транспортный", осуществляющий связь между системой и таким же аналогичным модулем, но функционирующем на другой ЭВМ.

В ДАКОМС предусмотрены языки нескольких уровней. Самый верхний уровень ориентирован на непрограммистов, однако имеет дополнительные операторы, позволяющие квалифицированным пользователям описывать сложные алгоритмы поиска , обработки, вывода. На одном из уровней система открыта к некоторым языкам программирования (ПЛ 1,АССЕМБЛЕР и др.) Внутренний язык системы - это язык "протоколов", предназначенный для поддерживания интерфейса между модулями системы. Движение информационных потоков подчиняется тем же принципам, как и протоколы в сетях ЭВМ.

Язык описания данных включает описания глобальной модели, схем, подсхем, локальных моделей (ВИО и ВТ), характеристик пользователей, полномочий для доступа, условий корреспонденции

(связанной с распределенной обработкой) и др. Разрешено описывать вторичные модели на основе глобальной модели, схем, подсхем, или уже описанных вторичных моделей. Эта возможность описания на нескольких уровнях разрешает некоторые проблемы, связанные с декомпозициями при распределенных системах. Трансляторы переводят описания данных в специальный внутренний формат, используемый модулями системы при работе с базами данных

К системе привязана одна системная база данных, содержащая набор стандартных процедур, описания данных, словари данных и элементов базы данных, информацию о секретности и защите, "почтовые ящики" и др. Эта информация создается и поддерживается либо администратором БД, либо пользователем, либо системой.

Почтовые ящики используются для "корреспонденции" и связывают систему и пользователи, подключенные к данной локальной системе, с всей глобальной системой (при распределенной обработке). Здесь можно указать и список процедур системы или потребителей, которые надо выполнять автономно и периодично. Синхронизация в описании данных и структуре таблиц осуществляется на основе этой части системной базы данных.

В системе ДАКОМС не предусмотрены собственные средства телеобработки, но обеспечен интерфейс с некоторыми интерактивными системами - ТСО, СИНТЕР и др. В этом случае система сама перенастраивается с пакетного на диалоговый режим работы и заботится о распределении и синхронизации ресурсов при коллективном доступе.

В конце 1981 г. были сделаны успешные эксперименты подключения некоторых существующих средств обработки данных к системе ДАКОМС. В настоящее время заканчивается разработка коммуникационного модуля для система ТОТАЛ. Результаты эксперимента и реальная експлуатация показали, что затраты на преобразование представления данных очень маленькие, т.е. такого рода интерфейс гетерогенных систем и моделей оказывается положительным.

ЛИТЕРАТУРА:

1) Hausel,B.C.,"A Unified Approach to Programm and Data Conversion",Proc.Int.Conf.Very Large Data Bases,Tokyo, 1977
2) Hausel,B.C.,"CONVERT: A High-level Translation Definition Language for Data Conversion",CACM,18,N 10, Oct. 1975
3) Shu,N.,Hausel,B.C.,"EXPRESS: A Data-Extraction,processing and Restructurings System",TODS,v.2, N 2,June 1977
4) Su,S.Y.W.,"Applications Programm Conversion Due tu Data Base Changes",Proc.Conf. Very Large Data Bases,1976
5) Su,S.Y.W.,"A Methodology of Application Programm analysis and Conversion Based on Data Base Semantics",Proc.ACM SIGMOD Conf., Toronto, 1977
6) Katz,R.H.,"Decompiling CODASYL DML into Relational Queries", TODS, March 1982,V.7,N 1
7) Katz,R.H.,"Database Design and Translation for Multiple Data Models", Ph.D. Desertation,Univ. California at Berkeley,1980
8) Taylor,R.W.,"Database Program Conversion: A Framework for Research",Proc.5$^{-th}$ Int.Conf.Very Large Data Bases,1979
9) Zaniolo,C.,"A Formal Approach to the Definition and the Design of Conceptual Schemata for Database Systems,TODS,v.7, N 1,March 1982
10)Lien,Y.E.,"Hierarchical Schemata for Relational Databases", TODS,v.6,N 1,March 1981
11)Katz,R.H.,"Heterogeneous Databases and High Level Abstraction",Proc.of the Workshop on Data Abstractions,Databases and Conceptual Modelling, Colorado,June 1980

# DATA INTEGRATION IN HETEROGENEOUS SYSTEM

A. Terziev

Abstract

A formal approach is proposed to the definition, integration and processing of data in heterogeneous database systems. Internal datamodel, access path graph and data transformation between different representation in database systems are concerned. In particular, a description of a heterogeneous database system is presented.

MTA SZTAKI Tanulmányok 147/1983 125-151

# SDLA
# SYSTEM DESCRIPTOR AND LOGICAL ANALYZER

E. Knuth, F. Halász, P. Radó

Computer and Automation Institute
Hungarian Academy of Sciences

## 1. GENERAL BACKGROUND

The major problem of information systems design methodologies is finding those adequate computerized tools which can support or replace the enormous clerical and organization work induced by the growing complexity of today's computer based systems to be implemented. Such tools usually consist of a "project-database", an "input-processor" which can interpret a certain descriptive language transforming it to computerized data, and a "query-processor". The latter may itself be pretty complex containing selection algorithms, facilities for logical analysis, mechanisms for formatted report generation, etc. Descriptions which are incrementally entering into the database can be checked with respect to various completness and consistence criteria. These criteria are usually also fixed in advance in accordance with the descriptive language selected. (The most succesful system of this kind has been the PSL/PSA [1] developed by the ISDOS project at the University of Michigan.)

When applying a particular computer aided methodology with a given descriptive language it frequently turns out however, that it is hard to match the predefined concepts of the language with those derivable from the problem we are faced. That is, the diversity of various application fields can not be easily captured by fixed languages constructed in advance. On the other hand, it turned out that any of the software tools supporting particular descriptive languages had essentially the same structure. It has been recognized, therefore, that it is possible to construct a set of computer aided tools independently of the application language and then these can be supplemented with those mechanisms facilitating language definitions.

In this way, so-called "meta systems" can be built providing the same capabilities as those of "convential" computer aided methodologies, but these are free of language restrictions for it is the user himself who designates his own conceptual world and language terms.

The approach of SDLA [2] is one of the possible ways to reach the goals described above. Another possible way is the one sug-

gested by the System Encyclopedia Management System (SEMS) [3] of the ISDOS project. (SEMS is based essentially on Chen's Entity-Relationship-Attribute model [4] while SDLA concentrates slightly more on semantic issues concerning its data model.) The development of SDLA has also been done in cooperation with the ISDOS project at the University of Michigan.

It should be noted that system SDLA is a set of fundamental tools for information systems design, but not a technology itself. The capabilities of SDLA are comprised in part 2. in this paper. SDLA can serve, however, as a computerized basis for various technologies. Part 3. outlines one of the possible application methodologies which can be suggested for a wide range of practical problems. Finally, part 4. demonstrates it by the Conference Organization Example using the features of SDLA.

## 2. FEATURES OF SDLA

### 2.1 System architecture

Figure 1. shown below recalls the simplified structure of conventional computer based supporting methodologies based on predefined descriptive languages.

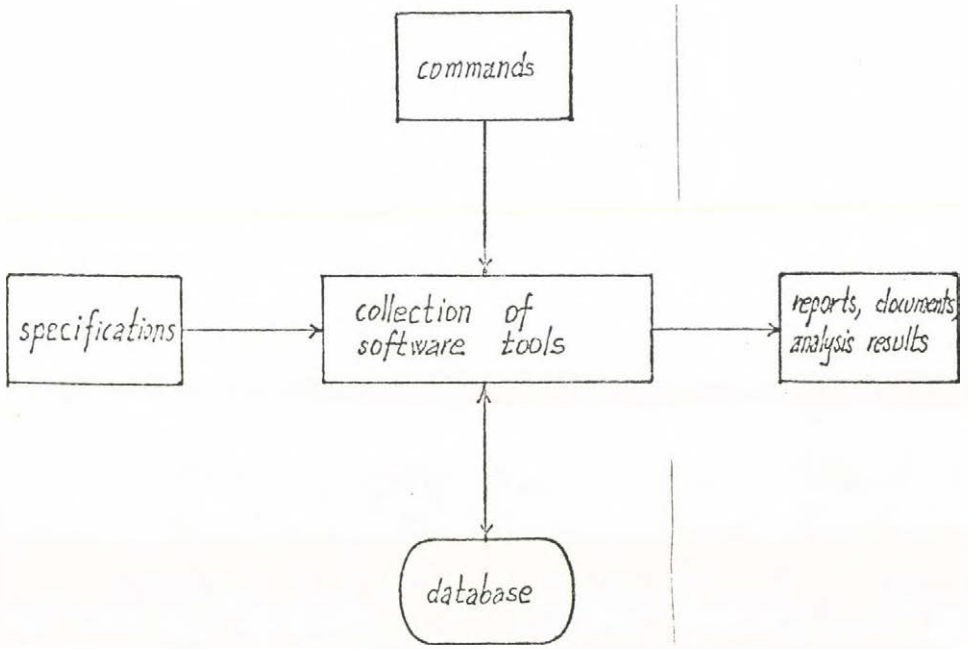



Figure 1.

The functional scheme of SDLA is slightly more compound as a consequence of the language independency, see figure 2. As it can be seen the software system is divided into two major parts namely the "language independent part" and the "language definition facility". From the viewpoint of realization the fundamental part is the former. This is built upon an appropriately chosen general scheme onto which each potential descriptive language can be mapped later. The "language definition facility" or "meta interpreter" is somewhat simpler, its major function is the generation

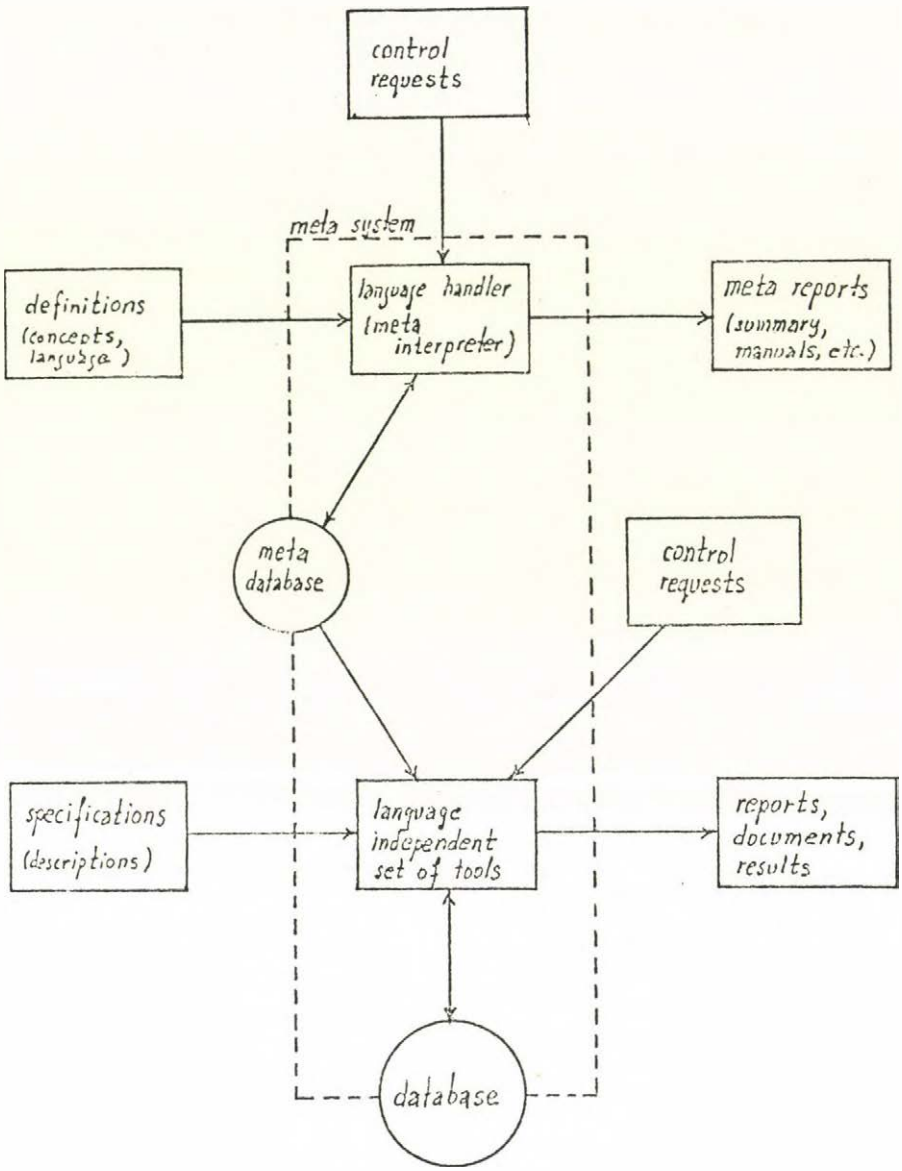



Figure 2.

of those tables which will control the operation of the language independent part. Besides, it have some additional functions too as the automatic generation of language manuals, reference cards, etc.

## 2.2 Internal data model

When it was found that a computer aid for system analysis may be developed without resorting to a particular methodology, it was also recognized that the computer aid should be developed based on a particular data model which would affect the way an area of real world problems is conceptualized (see also [5]).

The data model of SDLA is derived from SIMULA 67 [6] with slight extensions (and further additions for language definition purposes). The model is a simple uniform one having just one kind of data type termed "concept" (which may represent "entities", "relationships", or even "attributes" in Chen's [4] sense as well).

Our basic scheme is the following. In the data base we store objects, each of which is an instance of an abstract concept. Objects are described by attributes. An abstract concept is characterized by its associated set of attributes, to which the attributes of its instances correspond in their number and types. (The actual set of objects as instances to a given concept can always be considered as a relation i.e. the subset of the Carthesian product of the attribute value ranges. This viewpoint is useful, as it is known, for formalizing operations in a data base.)

Instead of going into syntactic details we illustrate the definition formalism of concepts by example 1.

```
concept modul;

concept data(part-of:data);

concept condition(associated-data:data);

concept precondition(to-activate:modul,when:condition);

concept postcondition(termination-of:modul,
                      resulting:condition);

concept invariant(condition,associated:modul);
```

Example 1.

For each concept (type) any number of subtypes can also be defined. Subtypes inherit all the attributes possessed by their supertypes, however, they may have extra attributes too, which are characteristic only for the special. Example 2. shows a collection of typical applications for this kind of type refinements.

The primary aim of the type refinement mechanism is capturing semantic aspects by a generalized version of type checking. Namely, during the instatiation at the descriptive level any special object is always accepted in a role of a more general. (For instance, any object of type "printfile" can participate a relationship declared for "files", however, "files", in general, may not attend any relationship declared explicitely for only "printfiles".)

```
concept process;

concept manual-process is process;
concept computerized-process is process;
```

```
concept file(opening:procedure,blocked:boolean);
concept printfile is file(pagesize:integer,line-
                                    length: integer);
```

```
concept input(process, data);

concept usage is input;
concept control is input;
```

Example 2.

We remark that each concept definition in SDLA can be supplemented with so-called "semantic constraints" and "integrity constraints" too. The former serves for automatic generation of statements derivable from the input descriptions, the latter ensures ways to declare compound checking considerations (being invariant throughout the whole life cycle).(These will not be detailed here, the reader can find it in [7].)

## 2.3 Language definition

Sentence form declarations constitute the second major function of the definition processor (meta interpreter). For each concept any number of "forms" can be attached which will tell us how the concept can be stated (and then instantialized) from contexts throughout the information system description. We illustrate this facility by example 3. only. For exact details we refer to the document [7].

```
concept data-derivation(process,used:data,derived:data);
    form used: used-by process to-derive derived;
    form derived: derived-by process using used;
    form process: uses used to-derive derived;
```

Example 3.

Declarations contained in example 3. tell us that whenever being in process context (process-section or process-subsection) during the description phase we may write the third sentence form, or when in a data context (data-section, data-subsection) we may use the first and second forms as well.

In this sense equivalent statements can be said in various forms as it is shown in example 4.

```
process P;
    uses D to-derive E;

data D;
    uses-by P to-derive E;

data E;
    derived-by P using D;
```

Example 4.

Whatever context form appears in the input text, the conceptual data representation is always the same, and the so called "total-report" will return it in all possible contexts (regardless of the origin).

## 2.4 Descriptive phase

as a whole is a serie of sentences entered incrementally and given by sentences stated in those forms declared at the meta level. They constitute sections and subsections of an arbitrary level of nesting. Example 5. shows a typical piece of a description of such fashion.

```
modul M;
    belongs to subsystem S;
    uses data D1,D2;
        to derive entity E;
            under condition C;
    uses data D3;
        via interface F;
    utilizes system WP;
data D4;
    created by M1;
    accessible for M2, M3 read only;
    etc.
```

Example 5.

Sentences in such a description are in juxtapositioned or in subordinated relations with each other shown by the positioning of lines. The positioning itself is made automatically by the input processor (controlled by a so called context-stack see [7]) which ensures a very important correctness checking facility (the so-called "context-checking") for the input descriptions.

A description like above is acceptable if all its subordination contexts correspond exactly to those declared at the meta form definitions, and object names appearing in the description coincides in their type (in the generalized sense concerning subtypes) to the type specifications of attribute definitions. (For more details we refer to [7].)

## 2.5 Formatted report system

is one of the important query facilities in SDLA. (We remark that as a consequence of the relational nature of the logical storage scheme a kind of relational query is also available. This, however, will not be detailed here. We refer to [7] concerning the heuristic descriptions and to [8] concerning its exact theoretical basis.)

The formatted report system communicates in the user defined descriptive language. Therefore, the user needs to know only one language (the very one defined himself). Example 6. shows such a report specification. This specification is acceptable if all

```
REPORT;
modul M;
  submodul any;
       dependent submodules any;
       related data any access any;
ENDREPORT
```

Example 6.

the sentence forms contained in it are declared at the meta level and their subordination contexts are proper. The symbols "any" (used in place of object names) are so called "free variables" which are to be filled according to the current contents of the database. (There are further possibilities too such as "named free variables", "set names" referring to previous selections, etc. For exact details we refer to [9].)

```
DOCUMENT
module M;
   submodule M1;
        dependent submodules A,B,X;
        related data D1 access immediate;
                    D2 access via interface F;
   submodule M;
        related data D3 access read only;
END
```

Example 7.

Example 7. shows a possible output concerning the report specification expressed in example 6. It demonstrates that the structure of the generated document corresponds exactly to the one given by the report specification, while free variables are replaced by actual names from the database.

The report generation facility illustrated by this oversimplified example above is substantial in the sense that

a) its results can be accepted as input descriptions too;

b) it is logically complete.

At the same time, the formatted report generation facility (together with the selection facilities not dealt with here) serves as a basis of further (more structured, more edited, etc.) reports, dumps, or graphic outputs generated by special routines.

## 3. DESIGN METHODOLOGY

The project database belonging to SDLA is intended to be used throughout the designed system's whole life cycle and, therefore, it can be asked or updated at any time later on. A full technology including all project management aspects too should refer to this whole cycle. In this paper we concentrate on its most crital phase starting with the interview process and ending with a complete logical system design which is sufficient for an average ability implementor to realize the system designed

### 3.1 Environment specification

Before starting the design process it is strongly advised to record a complete environment specification and enter it to the SDLA database. This description should include the following major parts:

A. Activity descriptions. Their characteristic data should be at least the following:

- who executes them;
- what sources are used;
- what do they produce;
- what kind of tools is utilized.

(Activities should be classified in the way shown in figure 3.)

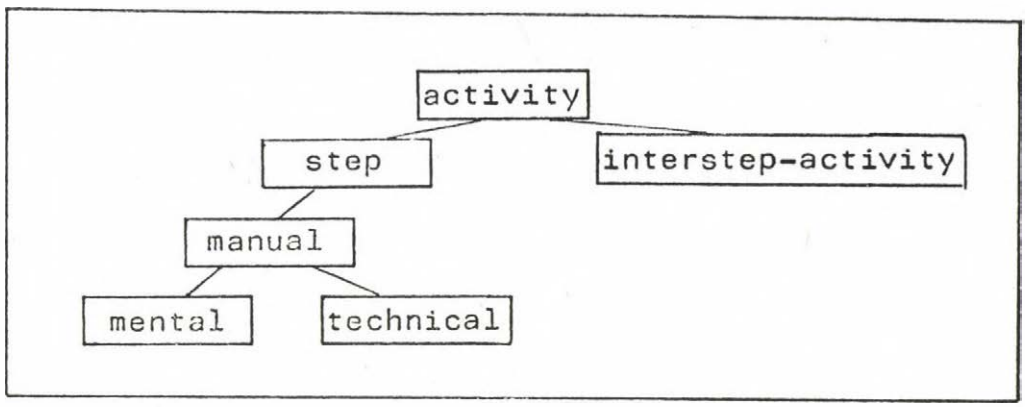



Figure 3.

The ordering among activities can be represented in several ways. In the simplest case serial numbers could be sufficient (as it is applied in our formalization of the Conference Organization Example in part 4. of this paper). In general, however, more complex models, Pert-graphs, Petri-nets, etc. can be used, see e.g. our proposals at [10].

B. Corporation description. This should include the organizational structure of the bodies, staff, personnel incorporated with the information system to be designed. The description should contain

- the activities they are involved;
- information they are using;
- documents they produce;
- their responsibility;
- their access limitations.

C. Description of documents. This part should concern to the content of those manual documents (vouchers, acknowledgements, protocols, minutes) which serve as a basis of the work of different bodies, and to those what they produce. In addition to the structure of the document the following details should be described

- who produces it;
- who uses it;
- which activities concern to it;
- access restrictions;
- validity limitations.

D. Description of data to be computerized. This description should made in the same way as of the manual documents. Note that in this phase we are not at all intersted in storage mechanisms, and data representation ways but concentrate only on functional structures!

E. List of tools which can be utilized by the participating personnel. This list can contain both kinds of devices which are available at present and which are to be implemented by the very information system being under development. The description need not cover other aspects than the overall function and structure.

Note that during the description process statements having the same logical content need not be said more than ones. E.g. if we have specified that during a certain activity a particular document is produced, then when that document is specified it is not necessary state its originating activity again. It is the function of the report system to structure and reproduce all implicite specifications automatically.

Note, also, that the process of description need not follow the list given above from A. to E. Information gained from the successive interview process can immediately be described formally and entered incrementally in any order. Again, it is the function of SDLA to group them systematically.

## 3.2 Logical design

Having the environment analysis finished, the so called logical design of the information system to be developed can be started. We use a top-down approach with successive refinements of the plan. At each step, during the design process, the system can be further decomposed until a final level. This final level should satisfy the following requirements:

i. The description still does not contain implementation details. Therefore, it may well be implemented by conventional file-systems, or by any kind of database management systems equally.

ii. It is complete in the sense that using the information contained in it an average programmer can easily realize it (or, what is the same, an average designer can easily produce its implementation-level plan).

The logical level system specification should cover the following major parts:

A. Information structures to be stored. Recall that we do have already data desriptions listed during the environment analysis. Now our task is to form top-down hierarchic structures from them in accordance with the hierarchy of program modules handling them. Data structures at the description may include the following aspects:

- type definition (set, list, element, etc.),
- inclusion relationships with other data,
- initialization, usage, update, and derivation relationships
- activization and timing conditions,
- access modes,
- access limitations and security considerations,
- restorability constraints,
- integrity properties.

B. I/O specifications. These are the data structures consumed or produced by software components. Their description should also be made by hierarchic refinement steps.

C. System functions. In this part software components (processes, procedures, routines, functional modules) are specified. The following data should be given:

- the function's input data,
- the function's output data,
- other data utilized,
- description of the function.

The process of successively refining descriptions is done in the following way. We start with a level-0 description which is usually the same for all information systems, see figure 4.

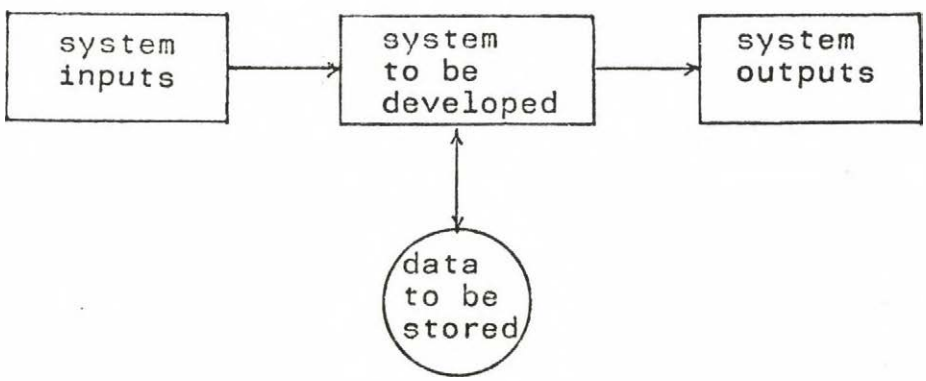



Figure 4.

Then all the four components shown in figure 4. are decomposed into a level-1 description containing newly specified relationships between the parts. Etc.

Note that interfaces receiving and generating I/O-s are embodied in the environment description. Remark, that final notes at point 3.1 are valid here too.

## 3.3 Implementation design

We only mention here that an implementation level system design

can be produced using SDLA but do not believe its necessity. The logical design should be fine enough to tell us all what is really functional. Implementation documentations should preferably made using standard utilities possessed by the implementation device applied. (Such documentation supports can be find in most up-to-date systems such as PROTE, METACOBOL, IDMS, ADABAS, etc.)
Anyway, it is possible to make implementation level descriptions by using appropriate concepts to represent storage mechanisms and program structures if needed. We do not go into this details here, the reader can find our proposals in the work [10].

## 4. THE CONFERENCE ORGANIZATION EXERCISE

This part illustrates the applications of the principles given in part 3. using the tools summarized in part 2. For better readability we disregard of starting with a boring sequence of sentence form definitions, but give the main parts of it as an appendix.

To obtain a better quality we do not give computer printouts here. The description written below is fully computer processable supposing the necessary definitions contained by the appendix is entered previously.

### 4.1 Environment description

A. THE ENVIRONMENTAL PROCESS

```
step 1 'p r o p o s a l' manual mental;
   made by WG;
   produces conference-organization-proposal;


step 2 'r a t i f i c a t i o n' manual mental;
   made by TC;
      using conference-organization-proposal;
   produces founding-document;


step 3 'c o m m i t t e e s   f o r m a t i o n' manual mental;
   parallel substep 'PC formation';
      made by PC-chairman;
         using conference-organization-proposal,
               founding-document;
      produces PC-member-list;
               PC-invitation-letter-form;
   parallel substep 'OC formation';
      made by OC-chairman;
         using conference-organization-proposal,
               founding-document;
      produces OC-member-list,
               OC-invitation-letter-form;
```

```
step 4 'd a t a b a s e   i n i t i a l i z a t i o n' manual
                                                    technical;
   device conference-supporting-system;
   actions;
      enter conference-head-information;
         into conference-database;
         based on founding document;
      enter PC-members,
            OC-members;
         into conference-database;
         based on PC-member-list,
                  OC-member-list;
      enter PC-invitation-letter-form,
            OC-invitation-letter-form;
         by word-processing;


step 5 'c o m m i t t e e s   i n v i t a t i o n' manual
                                                    technical;
   at request of PC-chairman,
                 OC-chairman;
   action;
      mail PC-invitations,
           OC-invitations;
         by electronic-mail;
         using PC-invitation-letter-form,
               OC-invitation-letter-form,
               PC-member-list,
               OC-member-list;
         utilizing word-processing,
                   conference-supporting-system;


step 6 'f i r s t   P C   m e e t i n g' manual mental;
   produces call-for-address-list,
            announcement-text,
            deadlines,
            minutes-PC1;
   made by PC;
      using founding-document;


step 7 'm a i l   o f   a n n o u n c e m e n t s' manual
                                                    technical;
   actions;
      enter deadline-data;
         into conference-database;
         based on minutes-PC1;
      enter call-for-list;
         into conference-database;
         based on call-for-address-list;
```

```
      enter announcement-text;
         by word-processing;
      mail announcement-text;
         by electronic-mail;
         using call-for-list;
         utilizing word-processing;
                   conference-supporting-system;


interstep activity from 9 to 15 'p r e r e g i s t r a t i o n'
                                          manual technical;
   made by conference-sectretariat;
      using conference-mail-received;
      utilizing conference-supporting-system;
   actions;
      update list-to-be-informed,
             list-to-participate,
             list-to-lecture,
             invitation-data;


interstep activity from 9 to 10 'p a p e r s
 r e g i s t r a t i o n' : manual technical;
   made by conference-sectretariat;
      using conference-mail-received;
      utilizing conference-supporting-system;
   action;
      update submission-head-information;


step 10 'd e s i g n a t i o n  o f  r e f e r e e s'
                                          manual mental;
   made by PC-chairman;
      using submission-head-information;
      utilizing conference-supporting-system;
   timed at submission-deadline;
   produces referation-designations,
            referation-viewpoints;


step 11 'm a i l  t o  r e f e r e e s' manual technical;
   actions;
      enter referation-information;
         into conference-database;
         based on referation-designations;
      enter referation-viewponts;
         by word-processing;
```

```
      mail paper-received,
           referation-viewponts;
         by electronic-mail;
         using conference-mail-received,
               reference-information;
         utilizing word-processing,
                   conference-supporting-system;


interstep activity from l1 to l2 'r e f e r e n c e
        r e g i s t r a t i o n' manual technical;
   made by conference-secretariat;
      using conference-mail-received;
      utilizing conference-supporting-system;
   action;
      update referation-information;


step l2 's e c o n d  P C  m e e t i n g' manual mental;
   produces acceptance-text,
            refusion-text,
            acceptance-list,
            session-designation,
            program-table,
            chairman-list,
            chairman-appointment-text;
   made by PC;
      using referation-information,
            submission-head-information,
            list-to-lecture,
            invitation-data;
      utilizing conference-supporting-system;


step l3 'a u t h o r  n o t i f i c a t i o n' manual technical;
   actions;
      update referee-information;
         using acceptance-list;
         utilizing conference-supporting-system;
      enter acceptance-text,
            refusion-text;
         by word-processing;
      mail author-notification;
         by electronic-mail;
         using referee-information;
         utilizing word-processing,
                   conference-supporting-system;
```

```
step 14 'c h a i r m a n   a p p o i n t m e n t' manual technical;
   actions;
      enter chairman-appointment-text;
         by word-processing;
      enter chairman-list;
      mail chairman-appointment-text;
         by electronic-mail;
         using chairman-list;


step 15 's e c o n d   O C   m e e t i n g' manual mental;
   produces list-of-accepted-participants,
            participants-reply-card-text;
            sorry-to-say-text;
   made by OC;
      using list-to-participate,
            invitation-data,
            call-for-list;


step 16 's e c o n d   a n n o u n c e m e n t' manual technical;
   actions;
      enter list-of-accepted-participants,
            program-table;
         into conference-database;
      enter participants-reply-card-text,
            sorry-to-say-text;
         by word-processing;
      mail program-table,
           participants-reply-card-text;
         by electronic-mail;
         using list-of-accepted-participants;
      mail program-table;
         by electronic-mail;
         utilizing ptc-selection-program;
            comment to those priority invited
            persons not intended to participate;


interstep activity from 16 to 17 'c a r d   r e g i s t r a t i o n'
                                                manual technical;
   made by conference-secretariat;
      using conference-mail-received;
      utilizing conference-supporting-system;
   action;
      update participant-list;
```

```
step 17 'f i n a l   d o c u m e n t s' manual technical;
   action;
      print participant-list,
            program-table,
            general-information;
         by conference-supporting-system;
```

B. DOCUMENT STRUCTURES

```
manual document conference-organization-proposal;
   parts topic,
         approximate-date,
         proposed-place,
         proposing-WG,
         PC-chairman-name,
         OC-chairman-name,
         reasoning;
   optional parts list-of-topic-subdirections,
                  list-of-PC-members,
                  list-of-OC-members;
                  list-of-participating-WGs;

manual document founding-document;
   parts conference-title,
         conference-date,
         conference-place,
         proposing-WG,
         PC-chairman-name,
         OC-chairman-name,
         discussion;

manual document PC-invitation-letter-form;
   parts conference-topic,
         conference-data,
         conference-place,
         PC-chairman-name,
         OC-chairman-name,
         date-of-first-PC-meeting,
         place-of-first-PC-meeting,
         polite-embedding-text;
```

C. DATA STRUCTURES

```
computerized data conference-database;
   part conference-head-information;
      containing qualification;
                 data,
                 place,
                 WG no,
                 TC no,
                 PC-chairman-data,
                 OC-chairman-data;
   part PC-members,
        OC-members;
      consisting of records of type personal-data;
   part deadline-data;
      containing date-of-submission,
                 date-of-notification,
                 final-copies-deadline,
                 participation-deadline;
   part call-for-list;
      consisting of records of type institution-address,
                                    personal-data;
   part invitation-data;
      consisting of records of type inv-pers-data;
      subsetting criteria priority-invited,
                          participates,
                          lectures,
                          rejects,
                          deputy-delegates;

   part list-to-be-informed,
        list-to-participate,
        list-to-lecture;
   part submission-head-information;
      consisting of records of type submission-entry;
   part referation-information;
      elements referee-data,
               paper-reference,
               return-date,
               qualification-data,
               textual-remarks,
               acceptance;
   part program-table;
      consisting of groups of type session-data;
      consisting of session-heading,
                    records of type lecture-data;
   part list-of-accepted-participants,
        participant-list;
   part reply-card;
      element name,
              hoteling-data,
              arrival-data,
              other-requests;
```

D. RECORD TYPES

```
record type personal-data;
   used in conference-database;
   elements distinction,
            name,
            status,
            affiliation,
            address,
            phone,
            telex;

redord type submission-entry;
   elements author-names,
            author-reference,
            title,
            institution,
            country,
            classification,
            qualification-data;

etc. ...
```

E. PERSONNEL

```
communities involved WG,
                     TC,
                     PC, PC-chairman,
                     OC, OC-chairman,
                     conference-secretariat;
```

F. COMPUTER AIDS

```
system conference-supporting-system;
   connected systems word-processing,
                     electronic-mail;

etc. ...
```

The whole specification given above can be entered into the SDLA database supposing that the corresponding concepts and sentence forms have adequately been given at the meta level. These are listed in appendix A.

Having the specification entered the database various reports can be produced using the selective report generator facilities. For instance the simple request

```
REPORT 'COMMUNITIES ACTIVITIES';
community any;
   makes step any;
END
```

will result in a document consisting of lists of steps grouped according to the communities making them. Or, as another example, the request

```
REPORT 'DATA HANDLING';
computerized data any;
   part any;
      entered by any;
      updated by any;
      used in any;
END
```

will list all computerized data together with their "entering", "updating", and "usage" associations.

## 4.2 Logical system design

According to 3.2 this part of the description deals only with the software complex to be implemented. Therefore, the specification and any document derivable from it by SDLA queries are dedicated to the implementors (in contrast to the environment specification which is dedicated to the designers).

As it is mentioned, the design is made by stepwise refinement of the scheme shown in figure 4. To decrease the extent we shall terminate only the main branches at the description.

### A. FUNCTIONS AND I/O RELATIONSHIPS

```
function conference-support;
    subfunction dedicated-functions;
        subfunction initialization,
                    PC-functions,
                    OC-functions,
                    mail-input-registration,
                    mail-output-automatism;
        end;
    subfunction standardized-mechanisms;
          subfunction word-processing,
                      electronic-mail;

function initialization;
     receives head-information,
              PC-member-list,
              OC-member-list;
     creates head-information-set;
     creates committees-subset;
          in personal-set;
```

```
function PC-functions;
    subfunction set-deadlines,
                enter-call-for-list,
                create-referee-set;
    end;
    subfunction referation-consumption;
          subfunction update-referee-set,
                      update-submission-set;


function mail-output-automatism;
    parameters name-of-text-to-be-mailed,
               set-of-addresses;
    utilizes word-processing,
             electronic-mail;


function mail-input-registration;
     subfunction registrate-first-reply-card,
                 registrate-papers,
                 registrate-referee-replies,
                 registrate-second-reply-card;


function registrate-first-reply-card;
     receives personal-data;
     receives desired-status-description;
            deduced by conference-secratariat;
                 based on first-reply-card;
     updates personal-status-information;
          in personal-set;


function registrate-papers;
     receives submission-head-information
     updates submissions-set;


etc. ...
```

The function decomposition process can be continued in the same way rather mechanically based on the environment specification. Having finished this design step the designer may ask some useful reports to support his further work. For instance, he may ask for a list of data which occured in the description entered so far. It will result in the following report:

```
REPORT

data head-information-set;
data personal-set;
   parts committees-subset,
         personal-status-information;
data submission-set;

   .
   .
   .
  etc.

END
```

This document can help the system designer when elaborating his data structures.

## B. INFORMATION STRUCTURES

```
set conference-information;
  subset head-information-set;
  subset personal-set;
       subset commitees-subset,
              call-for-address-set,
              invited-persons-set,
              submitters-set,
              referees-set,
              participants-set;
            subsetting criteria personal-status-information;
  subset submissions-set;



input conference-input;
    subpart head-information,
            PC-member-list,
            OC-member-list;
    end;
    subpart minutes-PC1-data;
          subpart deadlines,
                  call-for-list;
          end;
    end;
    subpart minutes-OC1-data;
           subpart invitation-list;
                subpart priority-invitation-list;
```

```
output conference-output;
    subpart address-list;
        subpart call-for-addresses,
                invited-addresses,
                submitters-addresses,
                referees-addresses,
                participants-list;
            subsetting criteria personal-status-information;
        end;
    subpart general-information;
        subpart headline,
                deadlines-designated,
                responsibilities;
    subpart program;
        subpart program-head-information,
                schedule,
                social-part;
```

C. DATA STRUCTURES

```
entity personal-data;
    contained in personal-set,
                 PC-member-list,
                 OC-member-list,
                 call-for-list,
                 invitation-list,
                 address-list;
    consists of  personal-identification,
                 personal-status-information;


entity personal-identification;
    consists of distinction,
                name,
                affiliation,
                address,
                phone,
                telex;


entity personal-status-information;
    consists of status-designations,
                status-functions;
  .
  .
  .
  etc.
```

Data elements can now be automatically derived from this kind of rough description by appropriate queries. The exact representation of data elements belongs to the implementation level plan. We do not detail it here.

## Appendix-A. Concepts for environmental description

This is a partial list containing only the main concepts. The reader could easily supplement it with the omitted details.

```
concept phase (name:text);

concept step is phase (no:integer);

concept manual-step is step;

concept manual-mental-step is manual-step;
   form absolute: step no name manual mental;

concept manual-technical-step is manual-step;
   form absolute: step no name manual technical;

concept computerized-step is step;

concept substep is step (part-of:step);

concept parallel-substep is substep;
   form part-of: parallel substep;

concept mental-activity(in:manual-mental-step);

concept making is mental-activity(personnel);
   form in: made by personnel;
   form personnel: makes step in;

concept source(information,of:mental-activity);
   form of: using information;
   form information: used by of;

concept production is mental-activity(document);
   form in: produces document;
   form document: produced at step in;

concept selection is mental-activity(information,
                                     from: information);
   form in: selects information from from;

concept information;

concept document is information;
concept manual-document is document;
concept computerized-data is information;

concept device(is:system,for: manual-technical-step);

concept action(in: manual-technical-step);

concept entering is action(in: action,entered:data);
   form in: enter data;

concept mail is action(in:action,document);
concept update is action(in:action,information);
concept print is action(in:action,information);

etc. ...
```

## Appendix-B . Concepts for logical design

(Partial list.)

```
concept function (part-of:function);
     form part-of: subfunction;

concept data-activity (function);
concept data-flow is data-activity;
concept reception is data-flow (input);
     form function: receives input;
     form input: received by function;

concept generate is data-flow (output);
     form function: generates output;
     form output: generated by function;

concept data-action is data-activity;

concept create is data-action (set);
     form function: creates set;
     form set: created by function;

concept update is data-action (set);
     form function: updates set;
     form set: updated by function;

concept deduce (reception ,staff);
     form reception : deduced by staff;
     form staff: deduces reception ;

concept basis (deduce,information);
     form deduce: based on information;

concept inclusion  (data-activity,data);
     form data-activity: in data;

concept prm-to-func (function,data);
     form function: parameters data;

concept utilize (function,utilized-function:function);
     form function: utilizes utilized-function;
     form utilized-function: utilized by function;

concept data;

concept set is data (part-of: set);
     form part-of: subset;

concept input is data (part-of:input);

concept output is data (part-of:output);

concept subsetting-criterion (data,is:data);
     form data: subsetting criteria is;
     form is: subsetting criterion for data;

concept entity (owner:entity);
```

etc.

References

[1] Teichroew, D. and Hershey, E.A. III, PSL/PSA a computer-aided technique for structured documentation and analysis of information processing systems, IEEE Trans. SE-3, 1(1977).

[2] Knuth, E., Radó, P., Tóth, Á., Preliminary Description of SDLA, Studies 105/1980, Computer and Automation Institute Hungarian Academy of Science (December 1979).

[3] Teichroew, D., Macasovic, P., Hershey, E.A. III, and Yamamoto, Y., Application of the Entity-Relationship Approach to Information Processing Systems Modelling, Proceedings, International Conference on Entity-Relationship Approach to System Analysis and Design, Los Angeles (December 10-12, 1979.)

[4] Chen, P., The Entity-Relationship Model - Toward a Unified View of Data, ACM TODS 1 (1976) 9-36.

[5] Teichroew, D., Knuth, E., Radó, P., Kang, K.C., Concept Refinement Approach (CRA), a System Specification Technique, CRA, Draft Paper, ISDOS Project The University of Michigan Ann Arbor (January 1981).

[6] Dahl, O.J. Myhrhang, B., Nygaard, K., SIMULA 67 Common Base Language, NCC (1970).

[7] Radó, P., Kiss, O., Knuth, E., Szilléry, A., SDLA 1.0 Users Manual, Working Papers II/14, Computer and Automation Institute Hungarian Academy of Science (November 1980).

[8] Knuth, E., Rónyai, L., Closed Convex Reference Schemes, Working Papers II/12, Computer and Automation Institute Hungarian Academy of Science (October 1980).

[9] Knuth, E., Radó, P., Halász, F., Formatted Report System of SDLA, Working Papers II/18, Computer and Automation Institute Hungarian Academy of Science (July 1981).

[10] Knuth, E., Radó, P., Principles of Computer Aided System Description, Studies 117/1981, Computer and Automation Institute Hungarian Academy of Science (November 1980).

MTA SZTAKI Tanulmányok 147/1983 153-161

# МАНИПУЛИРОВАНИЕ ДАННЫМИ В КОМПЛЕКСЕ ДИАС

Моника Христова Филипова
Институт математики с ВЦ, Болгарская Академия наук

Комплекс ДИАС поддерживает динамические производные ассоциации между данными баз данных СУБД ОКА [1]. Рассматриваются два типа производных ассоциаций: бинарное отношение между сегментами, исходным и зависимым типами, и динамический ключ, реализующий инвертирование в базе данных и переупорядочение экземпляров сегментов определенного типа в базе данных. Материализуются эти ассоциации при помощи служебных массивов-справочников, а именно, ассоциатора для бинарного отношения и индекса для динамического ключа. Индексы и ассоциаторы реализованы в комплексе как стандартные базы данных СУБД ОКА для служебного пользования. Комплекс обеспечивает создание и ведение справочников, реализующих динамические ассоциации, и языковый интерфейс пользователя с данными, использующий динамические ассоциации. Комплекс является сервисным средством для пользователя СУБД ОКА или системы МАКРОБОЛ [2], функционирующей как СУБД с включающим языком Кобол.

Предметом настоящей работы являются средства маницулирования данными в комплексе ДИАС. Эти средства позволяют пользователю строить Кобол-программы, взаимодействующие с СУБД ОКА при помощи операторов ЯМД, использующих производные ассоциации. Они расширяют ЯМД системы МАКРОБОЛ и при их реализации сохраняются все принципы организации обмена данными между прикладной программой и СУБД, принятые в системе МАКРОБОЛ, а именно:

- Данные, хранимые посредством СУБД ОКА, представляются пользователю в виде максимально приближенном к методологии и терминоло-

гии языка Кобол.

- С точки зрения пользователя база данных представляет собой логический файл, состоящий из записей различного типа, которые иерархически упорядочены в соответствии с упорядоченностью сегментов в СУБД ОКА. Каждому сегменту сопоставляется тип записи в логическом файле.
- Описание множества логических файлов, выделяющих подмножество данных, хранимых в базе, которое будет использоваться некоторым семейством Кобол-программ, представляет собой подсхему. В подсхеме данные описываются в терминах языка Кобол.
- Операторы ЯМД обращаются к данным по определенным в подсхеме именам логических файлов, логических записей или полей.

Для каждого логического файла подсхемы выделяется рабочая область Кобол-программы, называемая областью обмена информации между прикладной программой и СУБД. Описание всех необходимых областей обмена в секции рабочей памяти генерируется автоматически системой на основании хранимого описания подсхемы и параметров операторов ЯМД, используемых в прикладной программе.

- Для каждого логического файла подсхемы система автоматически генерирует описание области в секции связи (описание маски *PSB* в терминах СУБД ОКА). Эта область содержит информацию о характеристиках выполнения запроса к соответствующей базе данных и позволяет следить из прикладной программы за результатами работы СУБД.

Кроме того, при работе с производными ассоциациями пользователь не обращается непосредственно к служебным базам данных индекса или ассоциатора, и не описывает в своей программе никаких областей или операторов вызова, связанных с их использованием в программе. Служебные базы данных управляются комплексом ДИАС и пользователь "их не видит".

Языковые средства комплекса обеспечивают операции поиска записей логического файла, следуя определенному бинарному отношению, а также инвертированный поиск записей по значению динамического ключа. Они включают три типа операторов поиска:

- оператор ПОЛУЧИТЬ ПО СВЯЗИ;
- оператор УСТАНОВИТЬ;
- оператор ПОЛУЧИТЬ ПО ИНДЕКСУ.

1. Оператор ПОЛУЧИТЬ ПО СВЯЗИ

Предназначен для извлечения определенной зависимой записи логического файла, связанной в заданном бинарном отношении с указанным экземпляром исходной записи, в область обмена.

Оператор имеет один из следующих трех форматов:

Формат 1.

ПОЛУЧИТЬ ПО СВЯЗИ имя-ассоциатора [И СОХРАНИТЬ]
имя-записи-1 ДЛЯ имя-логического-файла

```
[ ЕСЛИ [ПЕРВАЯ    ]                                          ]
       [ПОСЛЕДНЯЯ  ] [ИСХОДНАЯ] имя-записи-2
       [ТЕКУЩАЯ    ]
       [ТИП        ]

[(имя-поля-1 операция-отношения-1 литерал-1
   [{И  } имя-поля-2 операция-отношения-2 литерал-2] ...)]
   [{ИЛИ}                                           ]
[ , [ПЕРВАЯ    ]                                          ]
    [ПОСЛЕДНЯЯ  ] [ИСХОДНАЯ] имя-записи-3
    [ТЕКУЩАЯ    ]
    [ТИП        ]
```

[(имя-поля-3 операция-отношения-3 литерал-3

[{И / ИЛИ} имя-поля-4 операция-отношения-4 литерал-4] ...)] ]]

Формат 2.

ПОЛУЧИТЬ ПО СВЯЗИ имя-ассоциатора [И СОХРАНИТЬ]
имя-записи-1 ДЛЯ имя-логического-файла
ЕСЛИ [ПЕРВАЯ / ПОСЛЕДНЯЯ / ТЕКУЩАЯ / ТИП] [ИСХОДНАЯ] имя-записи-2
КЛЮЧ литерал.

Формат 3.

ПОЛУЧИТЬ ПО СВЯЗИ имя-ассоциатора [И СОХРАНИТЬ]
СЛЕДУЮЩУЮ имя-записи-1 ДЛЯ имя-логического-файла.

Синтаксические правила.

а) операция отношения - это одна из операций = , 7= , > , >= , < , <= ;

б) все литералы,указанные во фразе ЕСЛИ,должны быть выровнены в соответствие с шаблонами полей и записываются по правилам написания литералов в языке Кобол;

в) имена записей во фразе ЕСЛИ должны принадлежать к одной и той же ветви иерархического дерева логического файла,содержащей сегмент,являющийся исходным в бинарном отношении.

Семантика оператора.

Имя-логического-файла и имя-записи-1 определяют тип извле-

каемой зависимой записи в бинарном отношении,определяемом именем-ассоциатора. Фраза И СОХРАНИТЬ указывается,если требуемая запись будет изменятся при помощи одного из операторов системы МАКРОБОЛ УДАЛИТЬ или ЗАМЕНИТЬ.

В форматах 1 и 2 фраза ЕСЛИ задает критерий отбора требуемого экземпляра записи исходного типа в указанном бинарном отношении. Этот критерий может выражаться условием для каждого типа сегмента ветви дерева,начиная от корневого и заканчивая требуемым исходным типом сегмента,как это принято в СУБД ОКА (Формат 1) или уникальным идентификатором сегмента - его сцепленным ключом (Формат 2). Заметим,что оператор ПОЛУЧИТЬ ПО СВЯЗИ для своей реализации требует доступа к записям двух логических файлов подсхемы. Имя одного из них (зависимого в бинарном отношении) явно указывается в операторе,тогда как другой логический файл,содержащий исходные записи,определяется системой из описания бинарного отношения. Для форматов 1 и 2 в результате выполнения оператора в область обмена помещается первый экземпляр записи зависимого типа, связанным указанным ассоциатором с экземпляром записи исходного типа,определяемым согласно заданному условию селекции.

При выполнении оператора формата 3 выдается следующий экземпляр записи зависимого типа (по отношению к предыдущему выполнению оператора ПОЛУЧИТЬ для указанного ассоциатора),что позволяет осуществить перебор всех зависимых записей,которые связаны с определенным экземпляром записи исходного типа в определенном бинарном отношении.

Варианты ПЕРВАЯ и ПОСЛЕДНЯЯ указываются если поиск записи должен начаться от первого или последнего экземпляра этой записи внутри текущего экземпляра физически исходной записи.

Вариант ТЕКУЩАЯ требует поиск,который начинается от текуще-

го положения на данном уровне. Вариант ТИП указывает,что если запись этого типа принадлежит к ветви текущей записи,то она удовлетворяет критерию отбора на данном уровне.

С помощью варианта ИСХОДНАЯ можно зафиксировать (в базе данных,которой принадлежит исходный тип сегмента в бинарном отношении) в качестве текущего выбранный экземпляр записи на этом уровне.

2. Оператор УСТАНОВИТЬ

Устанавливает позицию исходной записи указанного бинарного отношения на следующий экземпляр записи.

Формат оператора.

УСТАНОВИТЬ СЛЕДУЮЩУЮ ПО СВЯЗИ имя-ассоциатора.

Семантика оператора.

В результате выполнения оператора для бинарного отношения определяемого именем-ассоциатора,текущим экземпляром исходной записи становится следующий (согласно ассоциатора) экземпляр исходной записи по отношению к предыдущему обращению к ассоциатору. В область обмена запись не помещается.

3. Оператор ПОЛУЧИТЬ ПО ИНДЕКСУ

Предназначен для извлечения определяемой по значению динамического ключа записи логического файла в область обмена.

Формат 1.

ПОЛУЧИТЬ ПО ИНДЕКСУ [И СОХРАНИТЬ]
имя-записи ДЛЯ имя-логического-файла
ЕСЛИ имя-поля = литерал.

Формат 2.

ПОЛУЧИТЬ ПО ИНДЕКСУ [И СОХРАНИТЬ] СЛЕДУЮЩУЮ

имя-записи ДЛЯ имя-логического-файла
ЕСЛИ имя-поля.

Синтаксические правила.

а) литерал,указанный во фразе ЕСЛИ,должен быть выравнен в соответствие с шаблоном указанного поля и удовлетворять правилам языка Кобол;

б) имя-поля должно относиться к динамическому ключу для указанной записи.

Семантика оператора.

Имя-логического-файла и имя-записи определяют тип записи, экземпляр которой извлекается из базы данных с использованием индекса. Фраза И СОХРАНИТЬ указывается,если требуемая запись будет изменяться операторами системы МАКРОБОЛ УДАЛИТЬ и ЗАМЕНИТЬ.

Фраза ЕСЛИ задает используемый индекс и значение динамического ключа (Формат 1) требуемой записи.

Для формата 1 в результате выполнения оператора в область обмена помещается первый экземпляр записи,имеющий значение динамического ключа,равное заданному литералу.

Фраза СЛЕДУЮЩУЮ в формате 2 указывает на поиск следующего экземпляра записи в соответствии с текущей позицией в указанном индексе. Этот вариант оператора позволяет осуществлять перебор всех записей логического файла,имеющих определенное значение динамического ключа.

Комплекс ДИАС включает предкомпилятор,который обеспечивает:

- В секции рабочей памяти и секции связи генерируемой программы,конструирует описание всех областей взаимодействия прикладной программы и СУБД ОКА,необходимых для работы со служебными базами данных.

– В разделе процедур конструируется последовательность операторов ВЫЗВАТЬ, осуществляющая обращение к служебной базе индекса или ассоциатора и последующее обращение к информационной базе данных, для всех операторов ЯМД, используемых в прикладной программе.

В результате работы предкомпилятора над прикладной программой получается стандартная Кобол-программа, которую можно подать на стандартный Кобол-транслятор.

## Литература

1. Андон Ф.И. и др. Основные положения системы управления базами данных ОКА. – УСиМ, 1977, № 2, с.32-35.
2. Бабенко Л.П., Бобров В.А., Волкова Н.А., Лисюк Т.А., Мельник Л.А., Осташко Т.А., Рогач В.Д., Синяговская В.В., Филиппова М. Х., Ющенко Е.Л. МАКРОБОЛ – система поддержки прикладного программирования в среде СУБД ОКА. – В кн.: Первая всесоюзная конференция "Банки данных". Тезисы докладов, сек.3, Тбилиси, 1980, с.26-32.

# DATA MANIPULATION IN DIAS

M. Filipova

Abstract

In the present paper tools for datamanipulation in DIAS are considered. The proposed tools permit to construct Cobol programs which interface with DBMS OKA by DML operators using derived associations. They extend the system MAKROBOL DML and provide search operations, namely: search of logical records within a particular binary relation, and inverted search of logical records by the value of the dynamical key.

MTA SZTAKI Tanulmányok 147/1983 163-168

# ОБ ОДНОИ МЕТОДЕ КОМПОЗИЦИИ ТАБЛИЦ

Е. Живкова, Р. Лесева, Й. Денев
Институт математики с ВЦ,
Болгарская Академия наук

Существуют разные способы реализации функции обновления информации в базе данных [1,2] .

В первой части доклада предлагается метод для реализации процесса обновления. Во второй части рассмотрена программная реализация метода как функция СУБД БИСЕС и возможности, предоставляемые ею потребителям для обновления данных.

1. Метод композиции двух таблиц.

Пусть T $n$- арная таблица , $A=\{r_1, r_2, \dots, r_m\}$ - множество строк T, $B=\{d_1, d_2, \dots, d_n\}$ - множество столбцов T. Над элементами множества A можно определить упорядочение по составному ключу $K=\{k_1, k_2, \dots, k_e\} \subseteq B$ и заданному порядку $\omega_1, \omega_2, \dots, \omega_e$ , где $\omega_i \in \{LT, GT\}$ следующим образом: $r_s \preceq r_t$ если исполнено $x_i^s \ \omega_i \ x_i^t$ , где $K^s$ - ключ $r_s$ , $K^t$ = ключ $r_t$ и $i$ - минимальное целое число, для которого $x_i^s \neq x_i^t$ .

Метод композиции заключается в следующем:

Пусть T1 $n$-арная таблица, $A1=\{r_1^1, r_2^1, \dots, r_{p_1}^1\}$ - множество строк T1, $B1=\{d_1^1, d_2^1, \dots, d_n^1\}$- множество столбцов T1 и A1 упорядоченое по ключу $K1=\{k_1^1, k_2^1, \dots, k_{e_1}^1\} \subseteq$ B1

T2 $m$-арная таблица, $A2=\{r_1^2, r_2^2, \dots, r_{p_2}^2\}$ - множество строк T2, $B2=\{d_1^2, d_2^2, \dots, d_m^2\}$- множество столбцов T2 и A2 упорядоченое по ключу $K2=\{x_1^2, x_2^2, \dots, x_{e_2}^2\} \subseteq$ B2.

Пусть $K=\{x_{i_1}, x_{i_2}, \dots, x_{i_e}\}$ пересечение K1 и K2 такое, что $\omega_{i_1}^1 = \omega_{i_1}^2, \dots, \omega_{i_e}^1 = \omega_{i_e}^2$ , $e \leq min(e_1, e_2)$.

Строки таблицы T1 удовлетворяют строгому неравенству: $r_1^1 \prec r_2^1 \prec \dots \prec r_{p_1}^1$ для K.

Для таблицы T2 допустимо наличие строк со совпадающими значениями клю ча K: $x_1^2 \leqslant x_2^2 \leqslant \ldots \leqslant x_{\rho_2}^2$.

Над упорядоченными таблицами T1 и T2 можно определить операции $\cup$, C и $W$ следующим способом:

Операция $\cup$ представляет собой операцию обновления строк таблицы T1 при помощи строк T2 при условии равенства соответствующего ключа, т.е. $x_{i_1}^1 = x_{i_1}^2$, $x_{i_2}^1 = x_{i_2}^2$, ..., $x_{i_\ell}^1 = x_{i_\ell}^2$. При этом ключ обновляемой строки T1 не изменяется, а содержание остальных элементов строки является результатом применения арифметических действий над элементами обновляемой и обновляющих строк.

Для иллюстрации этой операции рассмотрим простые таблицы T1 и T2 на рис. 1, содержащие информацию о номерах поставщиков (S#), поставляемые номера деталей ( P# ) и поставляемые количества деталей (QTY ). Таблицы упорядочены по составному клю чу K={S#, P#} по порядку $\omega_1 = \omega_2 = LT$.

T1

| S# | P# | QTY |
|---|---|---|
| S1 | P1 | 100 |
| S2 | P1 | 10 |
| S4 | P2 | 50 |
| S5 | P1 | 30 |

T2

| S# | P# | QTY |
|---|---|---|
| S1 | P1 | 10 |
| S2 | P1 | 20 |
| S2 | P1 | 5 |
| S3 | P2 | 100 |
| S4 | P2 | 10 |

Результат

| S# | P# | QTY |
|---|---|---|
| S1 | P1 | 110 |
| S2 | P1 | 35 |
| S4 | P2 | 60 |
| S5 | P1 | 30 |

Рис. 1

Строка (S1,P1,10) из T2 обновляет строку (S1,P1,100) из T1, строки (S2,P1,20) и (S2,P1,5) из T2 обновляют строку (S2,P1,10) из T1 и строка (S4,P2,10) из T2 обновляет строку (S1,P2,50) из T1 путем складывания содержимого элемента QTY .

Операция C представляет собой операцию создания новых строк на основе строк T2 без изменения их ключей и их добавление к T1 таким способом, что для строк $r_1^3, r_2^3, \ldots, r_{P_3}^3$ результатной таблицы T3 исполнено: $r_1^3 \prec r_2^3 \prec \ldots \prec r_{P_3}^3$.

На рис. 2 приведен пример обновления таблицы T1 при помощи строк T2 при чем строки T2 добавляются к T1 без изменения.

T1

| S# | P# | QTY |
|---|---|---|
| S1 | P1 | 10 |
| S3 | P2 | 10 |
| S5 | P1 | 50 |

T2

| S# | P# | QTY |
|---|---|---|
| S2 | P1 | 5 |
| S2 | P1 | 100 |
| S3 | P2 | 15 |
| S4 | P3 | 20 |

T3

| S# | P# | QTY |
|---|---|---|
| S1 | P1 | 10 |
| S2 | P1 | 5 |
| S3 | P2 | 10 |
| S4 | P3 | 20 |
| S5 | P1 | 50 |

Рис. 2

Строка (S2,P1,5) из T2 добавляется между строками (S1,P1,10) и (S3,P2,10) из T1, а строка (S4,P3,20) из T2 между строками (S3,P2,10) и (S5,P1,50) из T1. Строка (S2,P1,100) из T2 не включается в таблицу T3 потому что ее ключ совпадает с ключом уже добавленной строки, а (S3,P2,15) из T2 не входит в T3 из-за того что ее ключ совпадает с ключом строки (S3,P2,10) из T1.

Операция W включает возможности обтовления строк T1 при помощи строк T2 ( как при операции ∪ ) и создания новых строк ( как при C ) и их обновление при помощи строк T2 ( как при ∪).

Рассмотрим пример на рис. 3.

T1

| S# | P# | QTY |
|---|---|---|
| S1 | P1 | 10 |
| S3 | P2 | 5 |

T2

| S# | P# | QTY |
|---|---|---|
| S1 | P1 | 20 |
| S1 | P1 | 30 |
| S2 | P2 | 10 |
| S2 | P2 | 15 |
| S4 | P1 | 100 |

T3

| S# | P# | QTY |
|---|---|---|
| S1 | P1 | 60 |
| S2 | P2 | 25 |
| S3 | P2 | 5 |
| S4 | P1 | 100 |

Рис. 3

Строки (S1,P1,20) и (S1,P1,30) из T2 обновляют строку (S1,P1,10) из T1 путем складывания содержимого элемента QTY . Строка (S2,P2,10) из T2 включается в T3, а (S2,P2,15) из T2 обновляет ее. Строка (S4,P1,100) из T2 включается в T3.

2. Программная реализация.

Метод композиции упорядоченных таблиц реализован как функция СУБД БИС$^{LC}$. Реализация основана на выполнении операций ∪, C и W во время слияния двух упорядоченных баз данных T1 ( обновляемая ) и T2 ( обновляющая ). Врамках одного заказа допустимо выполнение трех операций над записями баз данных. Каждая запись базы T2 содержит информацию для выбора соответству-

ющей операции. В заказе можно описать не более одной операции типа U , C и W , так что все записи , над которыми применяется одна и та же операция подвергаются обновлению, расширению или сжатию одним и тем же способом.

Созданые программные средства предоставляют пользователям возможности для:

- обновления существующей базы данных
- создания новой базы данных путем расширения и обновления существующей.

Функция реализована на языке Ассемблера ЕС ЭВМ.

ЛИТЕРАТУРА

1. Информационные системы общего назначения, Под ред. Е. Л. Ющенко, Москва "Статистика", 1975

2. Дж. Мартин, Организация баз данных в вычислительных системах, Москва "Мир", 1978

3. Документация СУБД БИСЕС, ИМ с ВЦ, БАН, 1980

# A METHOD FOR TABLE COMPOSITION

E. Jivkova, R. Leseva, J. Denev

Abstract

A method for composing tables is proposed. The rows of tables are ordered by a given composite key. Three binary operations are introduced which require a certain key correlation. A little example for each of them is supplied. On the background of these operations a method for data updating in a Data Base System is outlined. It is realised as a function of DBMS BISES.

MTA SZTAKI Tanulmányok 147/1983 169-174

# INDUSTRIAL COMPUTER AIDED INFORMATION SYSTEM BASED ON THE DISTRIBUTED DATA BASE MANAGEMENT (A CASE STUDY)

T. Remzső

Computer and Automation Institute
Hungarian Academy of Sciences

## 1. Introduction

One of the fastest developing trends in current computer technology is that of distributed processing, where the computing system consists not of a single processor with attached memories, but of a network of processors, possibly geographically dispersed, which communicate with each other over lowbandwidth communications lines. This environment raises the possibility of a *distributed data base*, where data may be *logically centralized* ( i.e. as viewied by the user ), but *physically decentralized* ( i.e. stored in separate segments at different nodes of the network. )

In a distributed data base, the data may be arbitrarily distributed over the nodes of the system and parts of it may be replicated at multiple sites. But the physical distribution and redundancy of the data *should be invisible to the user.*

We have three principal objectives of a distributed data base system:

- to achieve enhanced performance by enabling multiple sites to perform the data management function, and thereby *reducing communication costs and increasing parallelism;*
- to improve system reliability and survivability, by enabling the system as a whole to continue functioning *despite the failure of individual sites;*
- to allow for modular growth, accomodating in a graceful fashion both the growth of a database past local storage capacities and the introduction of additional sites to cope with increased processing requirements.

A distributed data base system must cope with *multiple copies of the same data* ( assuring their mutual consistency ), and with transactions operating *concurrently* at different sites. It must also provide *robustness* and *recovery* in the presence of site or communication failures.

The distribution context provides an especial motivation for work on *data translation*. Data translation is concerned with translating data from one format to another. This issue must be addressed in transferring data from one DBMS to another. In a computer network with different sites running different data base systems and data moving among them, effective data translation is critically important.

## 2. The Trade Company

The firm, where we must develope an information system is a trade company. The trade organizations are decentrally organized. The control of such organizations can be characterized as *decentral*. The nowadays informational provisioning is centralized and in this way not the appropriate organization form. The way to solve this contradiction is the *decentralization of data processing*. In connection with the decentralization of appropriate user processes this data decentralization leads to *different* data management or database management systems.

The most important tasks of this company are

to export
to import
to sale in Hungary

of electronic elements and parts. The firm deals with about 200 thousands of various items in one time.

The basic processes of the work in this firm are the follows:

- to receive orders from Hungary and abroad;
- to collect these orders;
- to make orders to Hungarians and abroads firms and factories;

- to store electronic elements - stock management;
- to dispose of these elements and stocks in hands;
- to give accounts and bills;
- book keeping;
- planning;
- data service to the ministry.

We have a nearly exact model of the organization of the firm. This model was the basis of our schemas ( logical data models ) in the central and the operational level of the distributed data base management system.

## 3. The Computer Network

The information system ( named "TEKER" ) is based on a *computer network* developed by the *Computer and Automation Institute of the Hungarian Academy of Sciences*.

The central machine of this network is an *EC 2035 computer* with 1 MByte CPU, With about 700 MBytes disc capacity and with 16 synchronous terminals.

The second level of this network is based on two Hungarian made *R-11 (VT 600, CM52) computers*. with 512 kBytes CPU, with about 100 MBytes disc capacity and with 24 asynchronous line terminals on each.

The computer network is based on a *programmed multiplexer system MS 790* developed in our Institute.

The elements of the software system at the central level are the follows:

OS/VS1 Operating System
IDMS Integrated Database Management System
DBMS
Integrated Data Dictionary
CULPRIT Report Generator System
On-Line Query System OLQ/2

SHADOW-2 Telecommunication Monitor

At the operation level the softare system has the following components:

MTM 2 Multifunction Monitor

FMS 2 File Management System

DMS 600 Database Management System.

## 4. The Information System

In this system ("TEKER") data collected from terminal users (tradesmans) is stored in *local storage units* and than transmitted in a block to the central computer and to the *central data base*.

The data may be divided into those

which can be stored *locally* and those

which are needed *centrally because of central control* ( for example master and financial data ) or because other users will employ them ( for example on line query system for partners and firm leaders. )

In the IDMS ( central ) database are all the data of the firm. In a DMS 600 database there is only data for a special trading area ( active elements, passive elements, electomechanical elements. )

The work of the central level's machine is mainly *batch* ( except the on-line query module. ) At the operational level the work of the computers is *primarily in on-line mode*. Developing and optimizing of data schemas we must take into consideration this fact.

The information system has three subsystems:

- Operating Subsystem
- Financial Subsystem
- Planning subsystem

In the Operating Subsystem ( at the operating level ) there are the following modules:

- basic data maintaining module
- order maintaining module
- disposing system module
- stock management module
- on-line query system for partners and leaders.

In the Financial Subsystem are the following modules:

- book keeping module
- price analyzing module
- bill making module.

In the Planning Subsystem there are the following modules:

- analyse module
- prognose module.

At the operating level are 48 user terminals for trading. The following data collections can be used by a trade specialist for his work:

- *stock in hands*
- *orders ( customer, deliver )*
- *expeditions*
- *some financial data ( e.g. rate of exchange, price informations )*

A tradesman has a special terminal oriented transaction collection ( written using the transaction language of the DMS 600 ) to navigate and maintain these data in DMS 600, to dispose wares, to make bill of delivers.

Master and financial data used for trading are coming from the central IDMS database to the second level DMS 600 database. These data are maintained using IDMS-COBOL programs. The data flow is at night from the central level to the operational level and from the operational level to the central level. Data are exact at the beginning of the trading session .

5. References

Gray, J., "Notes on Data Base Operating Systems,"
in Operating Systems: An Advanced Course,
Springer-Verlag, April 1978.

Rothnie, J.B. and Goodman, N., "A Survey of Research and Developement
in Distributed Database Management,"
Proceedings of the Third International Conference on Very
Large Data Bases, October 1977.

Zloof, M.M., "Query-by-Example: a data base language,"
IBM System Journal, N° 4. 1977.

## ИНФОРМАЦИОННАЯ СИСТЕМА ДЛЯ ПРЕДПРИЯТИЙ С РАЗДЕЛЕННОЙ БАЗОЙ ДАННЫХ

### Резюме

Настоящая статья представляет основные принцыпы разработки информационной системы с разделенной базой данных. Далее представляет управляющую систему предприятия, которому принадлежит информационная система, и сеть электронных вычислительных машин ЕС - СМ, элементы ее математического обеспечения, которые являются фундаментом для информационной системы.

# COMPUTER AIDED DESIGN (CAD) AND PERSONAL COMPUTERS

# ЦАД И ПЕРСОНАЛЬНЫЕ КОМПЬЮТЕРЫ

MTA SZTAKI Tanulmányok 147/1983 177-186

# ФУНКЦИОНАЛЬНЫЙ ПРОГРАММНЫЙ ЯЗЫК ДЛЯ КОМБИНАТОРНЫХ ИССЛЕДОВАНИЯХ С ПОМОЩЬЮ ЭВМ

Красимир Манев
Институт математики в ВЦ, Болгарская Академия наук

## 1. Введение

В [1] предложен проект системы для комбинаторных исследованиях с помощью ЭВМ. Туда же показано, что большая часть потребителей такой системы будут специалисты в области комбинаторики, которые никогда не применяли ЭВМ в своей работе и наверно и в будущее не смогут (либо не захотят) заниматься изучением какого нибудь конвенционального языка программирования. Это связано с трудностями с которыми сталкивается неискушенный в области ЭВМ специалист при попытке научить такого языка. В практике часто разрабатываются специализированные языки очень высокого уровня для таких специалистов. Эти языки однако не обладают большую гибкость.

В своей фундаментальной работе [2] Бэкус, подвергая критики недостатки классических языков фон Нейманового типа, ставит основы нового, функционального стиля программирования. Хотя некоторые идеи Бэкуса дискусионны, они дают возможность конструирования новых языков обладающие как простотой семантики, так и высокую гибкость. У языков функционального стиля еще ряд другие хорошие черты - легкая трансляция, возможность расширения и т.д., но здесь обсуждать их не будем.

В настоящей работе предлагается язык функционального типа, как язык запросов программной системы для комбинаторных исследованиях. Надо отметить, что это будет скорее идеологическое, чем формальное описание языка.

## 2. Основные понятия

Следуя [2] даем следующее

Определение 1. Функциональная система программирования определяется пятеркой $\langle \mathcal{O}, F, (), \mathcal{F}, \mathcal{D} \rangle$, где:

- $\mathcal{O}$ - множество объектов;
- $F$ - множество функций, отображающие объект в объект;
- $()$ - операция аппликация, которая связывает какую нибудь функцию из $F$ с каким нибудь объектом из $\mathcal{O}$ и обозначает результат выполнения этой функции над этим объектом;
- $\mathcal{F}$ - множество функциональных форм, позволяющие из элементов $F$ строить более сложные функции (процедуры);
- $\mathcal{D}$ - множество дефиниций, дающие имена процедурам, составленные при помощи $\mathcal{F}$ и объектам, к которым надо обращаться по имени.

Далее, обобщая известное понятие комбинаторики и информатики даем другое основное

Определение 2. Пусть $Z_p$ ( где $p$ - простое целое ) - конечное поле Галуа с характеристикой $p$, либо кольцо целых чисел (тогда его будем обозначат $Z_0$ ), $n$ - целое положительное число и $Z_p^n = Z_p \times Z_p \times \ldots \times Z_p$ ($n$ раз). Совкупность $\alpha_1, \alpha_2, \ldots, \alpha_m$ где $\alpha_i \in Z_p^n$, $i = 1,2,\ldots,m$ и $\alpha_i$ не обязательно различно от $\alpha_j$ когда $i \neq j$, будем называть обобщеной $n$-арной реляцией, эсли хотя бы один из параметров $n$ и $m$ не равен 1. Как обычно эсли $n = m = 1$, то такого объекта будем называть скаляром. Число $n$ называем степенью реляции, $m$ - ее кардинальность, а $p$ - характеристикой.

Определение 3. Пусть $A$ произвольное множество. Упорядоченная $k$-орка элементов $A$, где $k$-конечное целое, будем называть кортежом, элементы входящие в $k$-орке - атомами, а порядковый номер атома в кортеже - просто номером атома в кортеже.

3. Определение функционального языка

Телерь приступим к описанию элементов функционального языка программирования, который ориентирован на комбинаторные исследования - CORELAN (COmbinatorial REsearchs LANguage ).

3.1. Множество объектов

Пусть A - совкупност содержащая:

- обобщенные n-арные реляции;
- скаляры;
- пустой объект $\perp$ .

Множество объектов $\mathcal{O}$ языка CORELAN содержит все кортежи над совкупностью A . Отметим, что эсли хоть один атом кортежа пустой объект, то всего кортежа будем считать пустым обектом. Атомы кортежа разделяются запятыми. Номер атома соответствует его месту в кортеже, эсли не задан явным способом при помощи описателя *k , где k - целое и определяет настоящий номер атома. Например,

A*2, B*1 обозначает B, A ,
A*3, B*1 обозначает B, $\perp$, A , а
A*2, B синтаксически неправильный.

3.2. Множество функций F и операция аппликация ()

Каждая функция языка CORELAN сопоставляет объекту из $\mathcal{O}$ другого объекта из $\mathcal{O}$ и отличается своим именем от других функций. Имена функций языка четирисимвольные, где символы - буквы латинскогс альфавита или цифры и как обычно первый символ обязательно буква. Эсли объект над котором выполняется функция $\perp$ , то и результат выполнения тоже $\perp$ .

Операция аппликация связывает функцию и объект, над которым она будет выполняться. Результатом аппликации является объект, который получается после выполнения функции. Аппликацию будем записывать следующим образом:

[( ] объект ) имя-функции ,

где [ ] обозначают как обычно, что левая функциональная скоба может не быть записанной, когда от этого не пострадает семантика аппликации. Например вместо

( атом1,атом2 ) функция

можем писать

атом1,атом2 ) функция ,

но не можем вместо

объект1)функция1,(объект2)функция2)функция3

написать

объект1)функция1,объект2)функция2)функция3

потому что это существено меняет семантику записа.

Надо отметить особое место имя функции в аппликации нашего языка. Его записывание после знака аппликации, вместе с наличием явного указателя для номера атома в кортеже дают потребителю некоторые преимущества во время интерактивной работы с системой.

Множество функций - самая динамическая часть языка. Она будет меняться в зависимости от потребностей потребителя, от новых типовых задач и важнейших новых результатов комбинаторики. Рассмотрим основные группы функций и ограничимся несколькими примерами.

3.2.1. Функции арифметики

Сюда можно отнести два класа функции. Первый - это функции арифметических операций - PLUS , MINS, MULT и DIVD . Семантика этих функций такова: эсли атомы объекта из одного и то же $Z_p$ , то арифметика выполняется по модулю $p$ , иначе арифметика целочисленная. Эсли все атомы скаляры, то арифметические операции выполняются обычным способом, слева на право. Эсли в кортеже есть и реляция, то правила посложнее и иногда результатом является $\perp$ .

Второй клас - это функции вычисляющие скаляры из из-

вестных формул. Например FACT вычисляет факториел, NUTC - коэффициенты Нютонового бинома, STR1 и STR2 - числа Стирлинга первого и второго порядка и т.д. Как правило арифметика целочисленная.

3.2.2. Функции генерирования новых объектов

Большая часть этих функций инспирированы реляционной алгеброй и модифицированы таким образом, чтоб соответствовали понятию обобщеная реляция. Вот несколько примеров:

- функция UNIØ производит объединение всех строк входящих в объекте атомов, при этом одинаковые строки повторяются столько раз, сколько встречались до выполнения функции;

- функция INTS находит все строки, которые встречаются во всех атомов исходного кортежа отчитывая повторяемость;

- функция CART находит реляцию являющуюся Декартовым произведением строк атомов входящих в аргумент;

- функция PERM выполняет перестановку столбцов реляции;

- функция JØIN выполняет свертка де Моргана над реляциями аргумента;

- функция PRKT строит новый объект из некоторых столбцов аргумента и т.д.

Сюда можно отнести и другие функции не имеющие аналогов в реляционнной алгебре, но очень нужные для работы с комбинаторными объектами. Например:

- функция CRØS строит новый объект путем перекрытия двух или больше исходных атомов;

- функция TRAN строит нозую реляцию путем транспозиции строк и столбцов исходной реляции;

- функция CGEN расширяет реляцию путем добавления все новые строки,являющиеся циклическими вращениями исход-

ных строк;

- функция RGEN генерирует случайным образом реляции с заданными параметрами;

- функция LINØ строит линейную оболочку исходной реляции, эсли она определена над конечным полем и т.д.

3.2.3. Функции выполняющие обработки специфические для разных объектов

Это самая динамическая часть множества функций. Она сильно зависит от конкретного применения системы и для разных потребителей будет содержать разные функции. Поэтому мы укажем только несколько примеров, которых будем использовать для илюстрации.

Функция ØRTØ находит ортогональную реляцию для реляции представленной исходным объектом; Эта функция может быть выполненой над кодами и матроидами.

Функция SPKT вычисляет однострочную реляцию представляющую весовой спектр исходной реляции.

Функция ISØM проверяет наличие изоморфизма и дает в результат перестановку столбцов реализирующую найденного изоморфизма, иначе ⊥ и т.д.

3.2.4. Функции управления

Функция INDX для заданных начальной стоимости, конечной стоимости и шага строит однострочную целочисленную реляцию являющуюся подмножеством множества индексов.

Функция SETX строит из всех атомов исходного объекта однострочную целочисленную реляцию, являющуюся подмножеством множества индексов.

Функция ZEQU проверяет равна ли стоимость аргумента 0 и эсли да, то результат 1, иначе 0.

Функция NZEQ проверяет равна ли стоимость аргумента 0 и эсли да, то результат 0, иначе 1.

Функция INPT вводит с назначеного устройства ввода

значения аргумента.

Функция ØUTP выводит на назначеное устройство вывода значения аргумента.

Функция ÐCLS отводит поле в памяти для объекта с параметрами атомы аргумента в таком порядке - степень, кардинальность, характеристика и т.д. Эта функция может встречаться только в левой стороне дефиниции объекта (см. 3.4.).

Функция SAVE запоминает наименованный объект в архив системы, а функции FIND и ÐELT соответно предоставляют и выбрасывают именованный объект из архива системы.

3.3. Множество функциональных форм

Функциональные формы дают нам возможность строить из простых функций сложные (процедуры) и таким образом превращают простого набора функции в систему для програмирования. В языке CØRELAN допустимы следующие функциональные формы.

Суперпозиция функции состоит в том, что на месте каждого атома аргумента может быть поставлена любая функция. Семантика этой формы проста - функция внутреннего уровня выполняется и полученый таким образом объект становится атомом объекта, который является аргументом внешней функции. Суперпозиция изображается путем выписывания функции на месте атома в аргументе. Вот как при помощи суперпозиции можно построить арифметическую функцию $f(X,Y,Z) = X^2+Y^2+Z^2$ :

X,X ) MULT , (Y,Y )MULT, (Z,Z )MULT ) PLUS

Мультипликация функции состоит в том, что функция применяется ко всем заданным атомам аргумента, несмотря на то, что исходная функция определена только для одного атома. Семантика этой формы сложнее и очень зависит от функции. Мультипликация функции изображаем ставя специального символа @ непосредственно перед именем функции. Вот несколько примеров:

- 1,10)INDX)@FACT

вычислит кортеж из 10 атомов в котором $i$-ий атом равен $i!$ ;

- X,Y,Z)@INPT

вводит значения последовательно для X , Y и Z ;

- X,(1,9,2)INDX)@PRKT

создает кортеж из 5 атомов, $i$-ий атом является $2i-1$ столбцом реляции X .

Мультипликация аргумента состоит в том, что вместо одну функцию к данному аргументу аппликируем список функций и результатом является кортеж в котором каждый атом - результат выполнения соответствующей функции списка над аргументом. Мультипликация аргумента задается путем выписывания список функции в ломанные скобки. Например

7,8)[PLUS,MINS,MULT,DIVD]

вычисляет кортеж 15,-1,56,0 .

Форма условие строится из трех функции. Семантика следующая. Первая функция аппликируется к аргументу. Эсли результат выполнения 1, то к аргументу применяется вторая функция, эсли 0 - третья, иначе результат $\perp$ . Условие будем изображать ставя тройку функции в обычные скобки. Например, эсли X кортеж из двоичных скаляров,

X)(ZEQU,MULT,PLUS)

делает следующее - проверяет первый атом равен ли 0 и эсли да - дает результат 0, иначе - брой единиц в кортеже.

3.4. Множество де финиций

Множество де финиций дает потребетелю возможность присваивать имена объектам и процедурам во время сеанса. Таким образом к объектам и процедурам можно обращаться по имени. Де финиции будем писать следующим образом:

процедура=> имя-результантного-объекта

и процедура-> имя-процедуры . Имена должны быть уникальными для потребителя,который знаком системе своим идентифи-

катором. Кроме этого дефиниционный символ => является знаком того, что процедура должна быть выполненой. Нетрудно заметить что дефиниции дают возможность организировать рекурсивные процедуры (см. 4.), что дает потребителю довольно большие возможности.

4. Пример

Рассмотрим следующую задачу. Нам известны некоторое колличество порождающих матриц двоичных (25,8) кодов. Интересуемся есть ли у нас такая матрица, что соответствующий код имел минимальный весь хоть 10. Эту задачу решаем в языке CØRELAN следующей программой:

```
. 25,8,2)ÐCLS => SPEK
. )INPT)LINØ)SPKT,(1,9)INDX)@PRKT)@ZEQU->PRØC
  SPEK)(PRØC,ØUTP,PRØC)=> ⌴
```

Литература:

1. Манев Кр., О возможностях применении ЭВМ в комбинаторных исследованиях, Доклады конференции КНВВТ и РГ-11 по информационному обслуживанию научных исследований, София, 1982 (под печати)
2. Backus J., Can programming be liberated from the von Neumann style? A functional style and its algebra of programs , Comm. ACM, 21,8(Aug.1978),613-641

# A FUNCTIONAL PROGRAMMING LANGUAGE FOR COMPUTER AIDED COMBINATORIAL RESEARCH

K. Manev

Abstract

Backus defines a functional language as a set of objects, a set of functions that map object into object, an operation - application, a set of functional forms used to combine existing functions to form new ones and a set of definitions that assign a name to new functions. For needs of computer aided combinatorial research a similar language is created. Its five basic elements are described and some examples are given.

MTA SZTAKI Tanulmányok 147/1983 187-191

# АВТОМАТИЧЕСКОЕ ОТОЖДЕСТВЛЕНИЕ БОЛГАРСКИХ СЛОВОФОРМ БЕЗ ИСПОЛЬЗОВАНИЯ СЛОВАРЯ ОСНОВ

И. Ненова
Институт математики с ВЦ, БАН

При решении разного рода задач, связанных с автоматической обработкой текста /в частности в автоматизированных системах информационного поиска, при автоматическом составлении частотных словарей и др./, возникает проблема автоматического отождествления разных форм одного и того же слова. Несмотря на то, что болгарский язык обладает многими признаками аналитизма /например, отсутствие падежных форм/, он все таки имеет весьма развитую систему окончаний, с помощью которых образуются всевозможные формы одного слова /от основы болгарского глагола образуются 22 разные формы, от основы прилагательного - 9, от основы существительного - 4 или 5/.

Один возможный подход к решению задачи об автоматическом отождествлении состоит в последовательном сравнении словоформ в тексте с элементами заранее заложенных в памяти ЭВМ двух словарей - словаря основ и словария окончаний - с тем, чтобы при совпадении текстовой единицы с комбинацией двух словарных единиц, перейти от них к исходной форме соответствующего гнезда. Так как в системах обработки естественного языка нельзя предусмотреть все лексическое множество, правильное решение задачи в этом случае полностью зависит от состава и объема словаря основ.

Значительно более гибким является подход, который состоит в использовании алгоритма, анализирующего комбинацию конечных морфем и букв отдельной словофрмы. Этот подход опирается на конечное число разрешенных в языке сочетаний букв и морфей в конце слова и построенные на его основе алгоритмы являются более универсальными.

Естественным критерием для отождествления двух или более словоформ может служить совпадение тех частей словоформ, которые остаются графически неизменными в различных формах каждой из них. Тогда остальная часть словоформ содержит лишь морфемы, отражающие грамматические признаки /например, род и число существительных имен, лицо, число и время при глаголах и т.д./. В огромном большинстве случаев изолированная таким образом часть совпадает с той частью слова, которая в грамматике обозначается термином "основа". В некоторых случаях графически постоянная часть состоит лишь из очень малого числа букв и однозначное распознавание слова становится трудным, а иногда и невозможным. По традиции в таких случаях можно использовать положение о том, что отдельные слова имеют несколько /обычно 2/ варианта основы, которые при эвентуальном включении в словарь рассматриваются как носители одинакового лексического смысла.

На основе этих соображений можно сделать вывод, что сегментирование болгарских словоформ на основу и окончание является достаточным для целей автоматического отождествления. Это дало нам основание приступить к построению сегментирующей процедуры с использованием в качестве отправной точки лишь словаря морфологических элементов. Этот словарь, называемый далее словарем окончаний, составлен на основе анализа болгарского словоизменения и объединяет в себе 109 возможных окончаний болгарских словоформ. Сегментирующая процедура должна обеспечить правильное сегментирование произвольной болгарской словоформы на основу и окончание, т.е. при обработке словоформы процедура должна идентифицировать в ее конце некоторый элемент из словаря окончаний или установить, что данная словоформа представляет собой неизменяемое слово или имеет нулевое окончание.

Тот факт, что данаая словоформа формально оканчивается на комбинацию букв К, совпадающей с некоторым элементом словаря окончаний, не является достаточным основанием для выделения этого элемента как реальное окончание словоформы. Между К и О могут существовать 4 типа реляций:

1. |К| = |О| , т.е. К совпадает с действительным окончанием словоформы /машин-ата; К = О = ата/
2. |К| < |О| , т.е. К является только частью окончания, которое следует выделить /път-ищата; К = ата, О = ищата/
3. |К| > |О| , т.е. К включает в себя действительное окончание словоформы /ординат-а; К = ата, О = а/
4. О = ≠ , т.е. рассматриваемая словоформа является неизменяемой или имеет нулевое окончание, так что ее нельзя сегментировать /тичешката/.

Лишь первая из указанных возможностей отвечает правильному сегментированию словоформы. Ошибку второго типа можно избегнуть, используя специальную организацию словаря окончаний, в котором при реализации алгоритма обеспечивается последовательное рассматривание окончаний в порядке уменьшения длины. Таким образом идентифицирование данной комбинации букв в конце слова как окончание происходит после того как уже отброшены как возможные окончания все более длинные элементы словаря.

Возможность ошибки третьего и четвертого типа следует иметь ввиду при составлении так называемого списка примеров. Такой список ставится в соответствие каждому окончанию словаря возможных окончаний. Для составления этого списка на основе подходящего описания исследованы все болгарские словоформы, которые имеют в конце соответствующую комбинацию букв. В зависимости от того каково соотношение в группе между правильным и неправильным сегментированием при формальном отсечении данного окончания, каждому из 109 окончаний можно присвоить тип:

Тип 2 присваивается окончаниям, которые участвуют при образовании отдельных форм. Лиш, в этих случаях отсечение данной комбинации букв обеспечивает правильное сегментирование, а во всех остальных приводит к ошибке.

Тип 1 присваивается окончаниям, характерным для болгарского сло-

воизменения; отсечение соответствующей комбинации букв в конце словоформы обеспечивает правильное сегментирование на основу и окончание в огромном большинстве всех случаев и лишь в некоторых случаях приводит к ошибке.

Понятно, что для каждого окончания типа 1 легче перечислить все примеры ошибочного сегментирования, а для каждого окончания типа 2 - примеры правильной сегментации. Списки составлены на основе инверсного словаря болгарского языка и содержат около 7000 словоформ. Кроме отдельных словоформ в них могут быть представлены и части слов, т.е. использование некоторых словообразовательных зависимостей позволяет заменить группу словоформ их общей диагностической частью.

При разработке алгоритма используется большой объем лингвистической информации - грамматической - при определении окончаний, выделение которых обеспечивает приведение к общей форме словоформ с одинаковым лексическим смыслом, и грамматической и лексической - при составлении списков примеров. После обработки этой информации и составления списков определение окончания каждой словоформы является однозначным и формально можно описать его следующим образом:

Обозначим через $О_1$, $О_2$, ..., $О_к$ элементы словаря возможных окончаний, через $Т_р$ - тип окончания $О_р$, а через $С_р$ - список примеров при выделении окончания $О_р$. При обработке произвольной словоформы Д существует одна из следующих трех возможностей:

1. Существует элемент $О_р$, такой, что $Т_р=2$ и Дє$С_р$. Тогда $О_р$ - это окончание, которое следует выделить при сегментировании словоформы.
2. Существует элемент $О_к$, такой, что $Т_к=1$, словоформа Д оканчивается на $О_к$ и Дє$С_к$. Элемент с наибольшим числом букв среди всех элементов с этим свойством следует выделить как окончание словоформы Д.
3. Словоформа Д является неизменной или имеет нулевое окончание.

Процесс определения окончания произвольной словоформы можно описать коротко следующим образом: на основе отождествления конечной комбинации букв словоформы с элементом словаря возможных окончаний проводится гипотетическая сегментация. С помощью информации, которая содержится в соответствующем этому окончанию списке примеров эта сегментация подтверждается или определенным образом корригируется.

Процедуру, реализующую описанный алгоритм, можно использовать как компонент каждой информационной системы, которая обрабатывает данные в форме болгарского языка. С одной стороны это существенно уменьшает объем базы лингвистических данных /так как ключевые слова задаются не во всех возможных формах, а только в виде основ/, а с другой - увеличивает свободу пользователя, который имеет возможность использовать каждую из форм ключевого слова в его самом естественном виде. Эту процедуру можно использовать и самостоятельно для получения разнообразной лингвистической информации - выявление основ в данном тексте, автоматическое составление частотных словарей и др.

## AUTOMATIC IDENTIFICATION OF BULGARIAN WORDS WITHOUT USING DICTIONARY OF BASIC WORDS

I. Nenova

Abstract

The author described algorithm for automatic segmentation of Bulgarian word forms to be applied in information systems of various kinds which process a great volume of text information.

MTA SZTAKI Tanulmányok 147/1983 193-200

# LINGUISTIC PROCESSORS

R. Pavlov - G. Angelova

Institute of Mathematics with Computer Center
Bulgarian Academy of Sciences

The development of modern society is marked by a growing symbiosis between the human being and the computer. Now the efforts to reach the full capacity of computers lead to the necessity of developing the man - computer dialogue. The elaboration of adequate language tools allowing interaction with the computer within the framework of fragments of natural languages considerably enlarges the efficient and mass use of computers in an enormous number of application areas.

The realization of the man - computer dialogue in natural language involves at least two basic prerequisites: an appropriate algorithmicised formal description of the part of the natural language that plays the most essential role in human communication and the creation of computer techniques for representation and processing of the formal description.

Natural language have been an object of formal study for a long time; in the last few years, along with the development of artificial intelligence, one can list a number of program systems assisting and realizing the man - computer dialogue with various aims and to various degrees. As an illustration we can cite the machine translation systems; Schank's system; various parsers of natural language syntax (for example, concerning the German language - in the HAM-PRM system (Hamburg), Russian - an ATN parser of the Computer Centre of the Academy of Sciences of the USSR, Moscow); the dialogue information logical system DILOS (Computer Centre of the Academy of Sciences

of the USSR, Moscow), and so on. These systems use different models in natural language formal description. In spite of certain similarities in the approach chosen for the accomplishment of the man - computer dialogue in a natural language, there might be strong differences between separate systems due to different fragments chosen from a natural language, due to the type of formal description used, or, due to the level reached in the processing of that description.

The Bulgarian language is inflexional one (like the other slavonic languages), without cases, with strongly developed tense and preposition systems and with free word order.

The flexive nature of the Bulgarian language presupposes at least two obligatory levels of formalization: descriptions within the word-form and within the framework of the sentence syntax. Descriptions allowing an up-to-date computer processing of the Bulgarian language are being developed during the last few years - at the Laboratory of Mathematical Linguistics at the Institute of Mathematics with Computer Centre of the Bulgarian Academy of Sciences and at the Institute of Bulgarian Language of the Bulgarian Academy of Sciences. The Laboratory of Mathematical Linguistics is working on a formal model of Bulgarian language morphology covering the entire language system as it is described in normative grammars without restrictions imposed by the problem area. An appropriate formal description of the syntax of basic kernel constructions of the Bulgarian sentences is also being elaborated. These models of the Bulgarian language represent one of the initial prerequisites for the creation of systems for a man - computer dialogue in Bulgarian language.

The present material sets forth the principles for realization of a system for a man - computer dialogue in a natural language, developed at the Laboratory of Mathematical Linguistics, describes some acts aspects of its implementation and discusses certain results obtained in this field.

The described system presumes:

1. A formal description of the most essential part of the natural language;
2. An apparatus for the representation of knowledge relevant to a problem area;
3. A dictionary of all terms used in the chosen problem area; the dictionary is compiled with the help of specialists in the given problem area;
4. A management system for a data base or a program package (for example, a relational DBMS and the package BMDP - biomedical data processing), i.e. ready software products servicing users in a definite problem area and having as usual their own language for the description and of the problem area data and the techniques to process the data.

The described system is a superstructure over the software product mentioned above and performs the following tasks:

1. Translation from a language close to the natural one into the internal language for description or processing the data of the superstructured software product;
2. An effort to analyse the correctness of the user's request in terms of the problem area and messages when discovering mistakes;
3. In the case of unclear points in the user's request, when possible to define it more accurately in interactive mode.

A similar system will be termed as a linguistic processor (or a linguistics preprocessor) and is viewed as a dynamic system which can serve as a superstructure over various problem areas. At the Laboratory of Mathematical Linguistic processors on two levels are developed - for access of users - nonprogrammers to a relational data base in Bulgarian language and for acces of users - non-programmers to the tools of statistical analysis proposed by the program package BMDP.

When we say "a natural language" in the present material, we mean the following:

1. A fixed list of phrases that can be used in the man - computer dialogue is not assigned;

2. Restrictions are not imposed on the grammatic structure of the sentences the man enters into the computer;
3. It is assumed that the man communicates with the computer in a correct natural language.

The linguistic processor usess descriptions of the problem area (a structural description of the objects of the problem area and the connections and relations between them and the dictionary of the problem area terms). The relationship between the descriptions of the problem area and the separate parts of the linguistics processor is given on Fig. 1.

The dictionary of terms from the problem area is compiled by a professional in the chosen problem area.

Beside the lexicology of the problem area, the linguistic processor uses service lexicology - interrogative pronouns, phrases "equal to", "less than", "greater than", "or", "and", "so...so", etc.

The phrases the user enters into the linguistic processor in a natural language go to the block for lexical and syntactical analysis. In the process of analysis, the terms from the dictionary that can be met in the user's request are recognized. In the case of incorrect values (for instance, a literal of inadmissible type) a dialogue with the user ois carried out. The resulting internal representation of the input phrase is called an internal representation at level 1 and is independent of the particular natural language. It is a "mirror - image" of the user's phrase in the terms of the problem area. The word-forms from the user's phrase that do not belong to the dictionary of terms and to the dictionary of service vocabulary, are not processed.

The structural description of the problem area represent a connected semantic network. In the process of semantic analysis, the internal representation obtained at level 1 is checked from the point of view of connactiveness as a subnetwork of the structural description of the problem area. If the internal representation at level 1 is an incorrect structure, a dialogue with the user is performed.

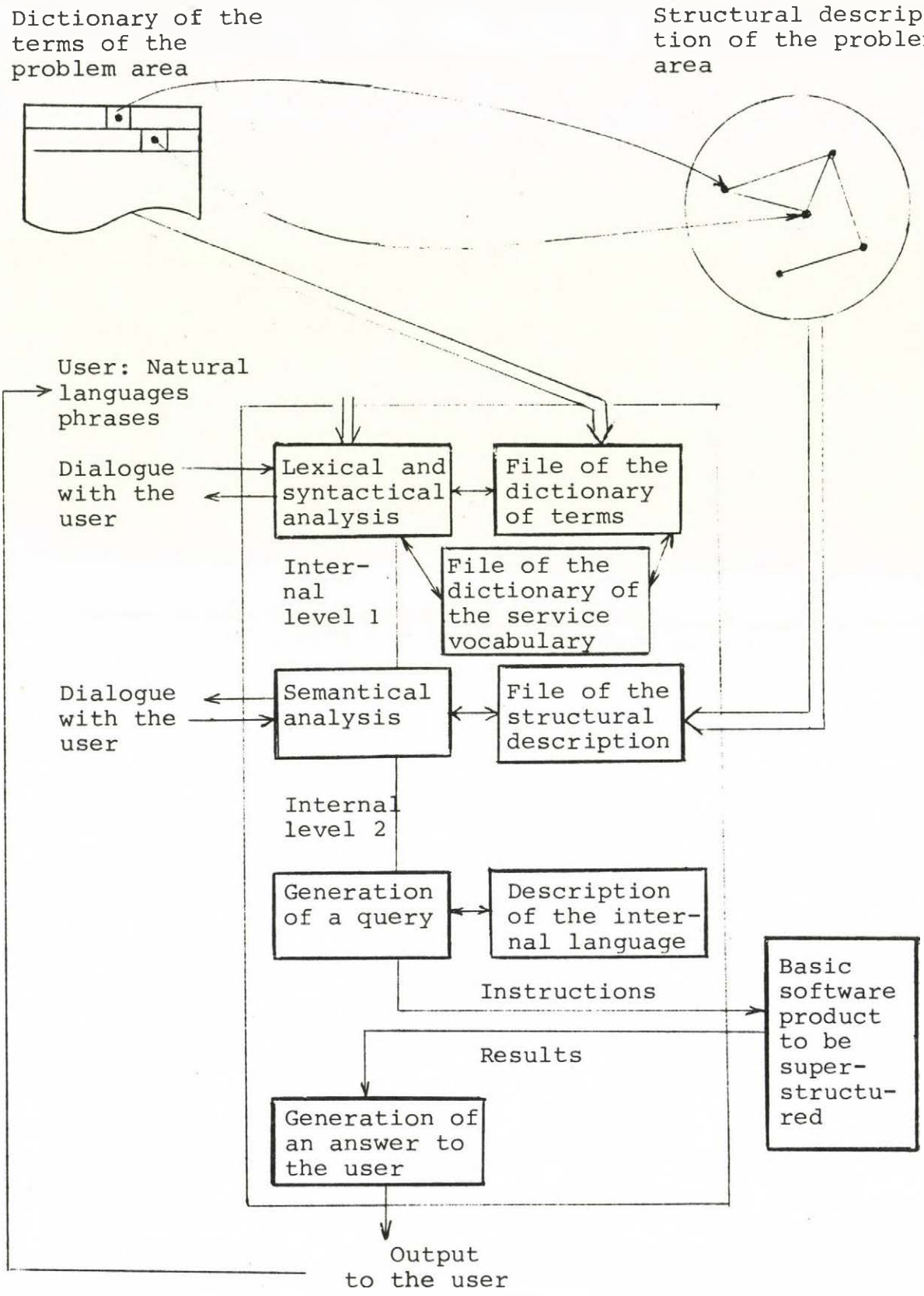



*Fig.1. A general scheme of the problem area descriptions and the separate parts of the linguistic processor.*

Provided the structure at level 2 is correct, the processor goes on to generation of instructions in the particular internal language of the superstructured software product. After these instructions have been transmited, the result is processed by the linguistic processor with the aim of synthesizing an adequate answer to the user.

Let us consider in more details the version of a linguistic processor allowing access of users - nonprogrammers to a relational data base management system. A query language similar to QUEL /1,2/ was chosen as an output language that the primary stage of the experimental implementation. The following reasons were taken into account:

1. The main aim was to investigate the translation process from the natural language question level into internal level two representation and to find suitable mathematical structures for description of this process as well well as processing algorithms.
2. The relational languages of this type offer certain simplicity of application: it is not necessary to show how the data are retrieved from the files of data base.
3. In comparison with the language ALPHA the chosen language is more comfortable because existencial and universal quantifiers are not required. It does not diminish the expressional capacity of the language, because in all real applications the relations are finite domains and due to this reason the predicate calculus is equivalent to boolean algebra.

The so called internal representation at level 1 corresponds to the following level of processing of the input natural language phrase: all "terminal values" /1,2/ and the values from the target list are recognized. All possible conditions of the type

file_name . attribute_name = 'literal'

have to be recognized during the translation from the natural language level into the internal level 1 representation. During the processing from internal level 1 representation to internal

level 2 representation the main problem is to complete the question in the chosen relational query language. Some conditions of the type

File_name_1. attribute_name_1 =
File_name_2 . attribute_name_2

have to be added to the internal level 1 representation. The algorithm for processing of the internal level 1 representation is described in more details in /4/. In order to accomodate the present version of the linguistic processor into a real DBMS certain changes are necessary.

A variant of a linguistic processor for the program package DBMDP has been developed at the Laboratory of Mathematical Linguistics. The results lead to the conclusion that the absence of a precise structural description of the problem renders difficult the elaboration of an algorithm for decision taking in various cases. At present, a detailed structural description of the problem area of the program package BMDP is being worked out in view of improving the operation of the linguistic processor related to it.

An aspect of interest is the application of this scheme for the man - computer dialogue in the field of creation of high - level languages for robot control.

In our opinion, the scheme applied for the realization of a linguistic processor for access of users - nonprogrammers to data base management systems or program packages in a natural language is a promising approach for the accomplishment of the man - computer dialogue in natural language in given problem areas.

REFERENCES

[1] Held, G.; Stonobraker, M.; and Wong, E. "INGRES - a relational data base system", in Proc. AFIPS 1975 National Computer Conf., AFIPS Press, N.J.

[2] Wong, E.; Youssefi, K. "Decomposition - a strategy for query processing", Technical note 78, University of Berkeley, Berkeley, California, 1975.

[3] Pavlov, R., Angelova, G. "On an approach for designing linguistic processors". Proceedings of the Second World Conf. "Mathematics at the service of men", Spain, July 1982.

[4] Angelova, G. "The use of natural language as a query language in a relational data base". Prcceedings of the IV International Seminar cn Data Bases, Schwerin, GDR, December 1981.

MTA SZTAKI Tanulmányok 147/1983 201-204

# ФОРМАЛЬНОЕ ПРЕДСТАВЛЕНИЕ ДИАЛОГОВЫХ ФОРМ В ИНТЕРАКТИВНЫХ СИСТЕМАХ

А.Л. Петков
Институт математики с ВЦ, Болгарская Академия наук

По формальному представлению диалога существует сравнительно мало публикаций. В /2,3/ основой для представления формального диалога используется структура абстрактного автомата. Другим подходом является предложенное в /4,5/ формальное представление посредством ориентированных нагруженных графах. Недостаток первого способа в том, что формальная модель не отражает структуры диалога как совокупность разных элементов, пока второй подход дает одно статическое описание структуры диалога,при котором нельзя проследить функционирование диалоговой формы.

В настоящей работе сделана попытка формально описать связи между элементами в диалоговой форме посредством абстрактной структуры, называющейся диалоговой сетью.

На базе основных определений из теорий сетей Петри /1/ предлагаем следующее определение :

Д И А Л О Г О В О Й С Е Т Ь Ю будем называть пятерку

$$\mathcal{DN} = (B, E; F, V, M_0)$$ ,где

а) $\mathcal{N} = (B, E; F)$ есть ориентированная сеть представляющая собой систему условий-событий.

E есть множество переходов событий

$$E = SU\mathcal{D}UMULUCUR$$

б) $V: F \to \mathcal{N}$ представляет собой набор конкретных правил для диалоговых сетей $\mathcal{DN}$, при помощи которых определяются способы перехода из едной маркировки $\mathcal{M}$ в другую маркировку $\mathcal{M}'$ на $\mathcal{DN}$.

в) $\mathcal{M}_0$ есть начальная маркировка для сети $\mathcal{DN}$.

г) некоторая маркировка $\mathcal{M}$ может перемениться при следующих правилах для переходов:

Если $e \in E$, $\bullet e = \{a_1, a_2 \dots a_m\}$, $e\bullet = \{b_1, b_2 \dots b_n\}$ будем говорить, что событие разрешено, если во всех входных условиях $a_i$ ,для которых $V(a_i, t) = and$ имеет по одному признаку,

хотя в одном из всех входных условий $a_j$, для которых $V(a_j,t) = or$ имеет признак,
во всех входных условиях $a_k$, для которых $V(a_k,t)=\overline{and}$ не имеет ни одного признака.

Когда осуществится переход, признаки выводятся из всех входных условий $a_i$ и переводится в выходные условия $b_j$ при котором в каждом $b_j \in e\bullet$, для которого $V(t,b_j)=i$ ставится по одному признаку, а из всех $b_1, b_2 \dots b_r$ условий, для которых $V(t,b_j) \asymp p_e$ для $i=1,2,\dots r$ $l \in \{1,2,3\dots\}$ признак ставится в одном из них или больше в счете $l$ из них. В случае правила перевода признаков типа $p$ необходимо соблюсти следующее условие: $l \leq r \leq n$.

В соответствии с предложенным определением одну диалоговую форму можно представить формально при помощи конкретной диалоговой сети $\mathcal{DN}$, соблюдая следующие конвенции:

а) каждый элемент диалоговой формы представляется как одно событие $e_i$ в диалоговой сети $\mathcal{DN}$, а в зависимости от типа элемента соответствующее ему событие принадлежит к одному из перечисленных подмножеств множества $E$.

б) множество входных условий $\bullet e_i$ для данного события определяется правилами поведения елемента в диалоговой форме и его взаимоотношениями с остальными элементами в этой же диалоговой форме. Правила поведения элемента в диалоговой форме отражаются и при определении типа входных дуг $(and, or, \overline{and})$.

в) множество выходных условий $e_i\bullet$ для данного события определяется типом и атрибутами соответствующего элемента в диалоговой форме.

г) вводится дополнительное подмножество событий $R$, внутренных для диалоговой формы, которые не отражают непосредственно данного элемента формы, а определяют некоторые специфические для конкретной форме переходы между условиями, которые переходы можно толковать и как события-реакции системы для данных условий. В подмножестве $R$ участвуют все те события, наступление которых означает конец диалоговой форме и соответственно акту взаимодействия. Совершение одного такого события включает соответствующий акт вычисления в рамки интерактивного вычислительного процесса.

Формальное представление диалоговых форм при помощи предложенных сетей создает следующие преимущества:

- сеть можно рассматривать как модель диалоговой формы, в которой отражены как структура, так и поведение формы. При проигрывании этой модели генерируется различные диалоги, допустимые для данной диалоговой формы, и можно определить их непротиворечивость.

- предложенная сеть относится к классу сетей, представляющих собой системы условий-событий, по которым развиты подробные теоретические методы исследования. В этом аспекте чтобы диалоговая форма была непротиворечива,необхдимо соответствующая ей сеть быть безопасной и устойчивой при заданной начальной маркировке.

- предложенные диалоговые сети могут стать хорошей методологической основой для изучения самых общих свойств рассматриваемого класса диалоговых форм и для создания специализированного языка для их описания.

## Литература

1. Genrich H., Stanhiewicz E.,"A dictionary of some basic notions of Net Theory", Simp. on General Net Theory, Hamburg, 1979.
2. Kupka I., "A structured Model for Dialog Languages", Lect. Notes in Econ. and Math. Syst., 75, 1971.
3. Kupka I., Wilsing M., "Functions describing interactive prcgramming", Proc. Int. Comp. Symp., NHPC, 1973.
4. Laguente J.M., "The Specifications of Data-directed Interactive User-Computer Dialogues", Ph.D. Thesis, Cornell Univ., 1977.
5. Laguente J.M., Gries D., "Language Facilities for Programming User-Computer Dialogues", IBM J. Res. Dev., 2, 1978.

# A FORMAL DESCRIPTION OF DIALOG ITEMS IN INTERACTIVE SYSTEMS

A.L. Petkov

A formal description of dialog items in man-computer interactions is proposed in this paper. It's based on a General Petri Net Theory. An appropriate substantiation within a generalized Petri Net is realized. It corresponds to specific dialog considerations. Advantages of a proposed formal description are presented.

MTA SZTAKI Tanulmányok 147/1983 205-228

# THE DESCRIPTION LANGUAGE OF THE CARS SYSTEM

M. Bohus, Gy. Csopaki, A. Filp
Technical University of Budapest

A. Hinsenkamp
Institute for Coordination of
Computer Technics, Budapest

L. Máté
Computer and Automation Institute
Hungarian Academy of Sciences

## Abstract

The system and language CARS applies a top-down structured approach in digital synthesis. The design language is well suited to describe behavioral concepts and structures as well. Refinement work of designer is supported by verification and testing procedures. Levels of design are neither predefined nor restricted. Test extension keeps the design process consistent while domain tightening assists to avoid combinational data explosion. Design steps are well documented and results are properly archivated.


## 1. Introduction

Fast growing complexity of digital systems to be designed increased the demand for computer aid in all phases of the design process [3,5]. The roughly 20 year history of CAD answered this challange with a wide spectrum of solutions in the routine field such as for example layout design for PCB and IC masks or test generation for SSI and MSI circuits etc. In higher spheres of logical design results are less spectacular, the role of computer in synthesis is rather modest: Investigations have been made and results achieved in simulation of hardware units for early detection of design-errors but results are not widespread in practice. The popularity of simulation methods hasn't increased more swiftly due to some reasons:

- restrictions in language /e.g. predefined levels of description like gate, register,etc., or separate handling of data flow and control/.

- operating of the simulated unit is possible only at a lowest level /gates/. A large system can't be described at gate-level because of memory capacity and speed--reasons.
- notions, structures and relations of simulation are not familiar to development engineers, special training and knowledge is needed for the user;
- high efforts needed on user-side for application /efforts for inputing, i.e. describing a system sometimes separately for every level are not proportional to the information achieved from the simulation/;
- lack of contingency in the CAD system between functional or logical description and the constructional phase of design process /layout design, firmware/.

What can CAD people do for the designer in the phase of synthesis? In our view we should support the designer's work with a very flexible system which gives him assistance when formulating his own ideas, syntactically and semantically controls the concepts developed, documents and archivates every detail of the design process and transforms the results automatically into the input forms of the existing CAD systems. Thus, making the whole synthesis process thoroughly checked well documented, easily modifiable and the results automatically transferred to the constructional phase we can assist the designer to keep track with the growing complexity of the synthesis task without restricting his skill and phantasy in the solutions developed and still easing the burden of his work with an almost continuous control.

## 2. Requirements

CARS project was started to gain a user oriented system according to the above philosophy. For this purpose the following requirements must be met:

- there should be only one language in the system which is suitable both for describing behaviour of a unit as a black box at any level and defining its structure by components /smaller level black boxes/ and their interconnections;
- design levels should be defined by the user, there should not be any predefined compulsory level on system-side. This provides for identity in structure of construction /functional or constructional levels and their hierarchy/ and in that of description;
- mechanisms and procedures provided should be totally level independent;
- only two neighbouring levels of the design should be taken care of in each design step: the level of behaviour and that of realizing structure;
- a structured top-down approach should be preferred to the bottom-up one, however, the later should also be supported;
- each step of the design process should be documented and results of the design should be archivated,
- direct interface should be provided for the CAD system of the constructional phase of the design process.

## 3. How to design in CARS?

The design process in CARS consists of three steps to be repeated recursively:

- user describes expected behaviour of Unit in question. This description is called "TYPE", and refers only to IN- and OUTPUTS of Unit without reference to structure. CARS checks description for syntax.

- user describes proposed realizing structure of the same unit using /expected/ behaviour of components and interconnections among them. CARS checks description for syntax,
- CARS compares structure to behaviour.

If structure fulfils requirements the same cycle may be repeated for all components of the structure at next level. Type descriptions exist already, they are to be derived from the previous level. Please note: fulfilling the requirements means a relation 'greater than' and not equality. The type in question may otherwise perform a lot more functions than specified at the given place of the structure. Necessary domain tightening can be achieved this way by neglecting unused functions and combinations.

When designing a digital system with CARS the designer defines the behavioral description of the whole system first. This highest level behavioral description $F_1$ functionally specifies the desired behaviour of the whole system. CARS checks the description $F_1$ automatically with respect to completeness and consistency. To control the functioning of $F_1$ the designer defines the input data $D_1$ and executes the simulation $Z(D_1, F_1)$ which gives the result: $P_1$. Evaluating these results the designer decides whether the behavioral description $F_1$ is right or not. If it is right a new type $T_1$ has been created by the behavioral description $F_1$ and is archivated in the CARS library together with the data $(D_1, P_1)$ which are called the definition tests for $T_1$. In the CARS approach the system to be designed is a copy $E_1$ of the type $T_1$. It is necessary to distinct between type T and it's physical realization E at a defined place. The same type may be used for more places in a structure /e.g. an IC/ refering always to the same functional description /test set/ but having different I/O connections.

Nevertheless, should the designer be not satisfied with the simulation results the whole step should be either repeated or the type $T_1$ should be modified.

Analysing type $T_1$ the designer can find that there is a realization $R_A$ stored in the CARS library which is able to realize the type $T_1$. He controls this statement by running the simulation $Z(D_1, R_A)$ resulting $Q_A$. Comparing $P_1$ and $Q_A$ he can prove that his estimation was right and in this case the design process has been finished.
Otherwise he finds that there is no realization stored in the CARS library which is able to realize $T_1$. In this case he designes the structure $S_1[E_j]$ which is intended to realize $T_1$. Structure $S_1$ consists of copies $E_j$ of types $T_i$. All the $T_i$ types should be defined by their behavioral description $F_i$ as it was the case with $T_1$. The proper interconnection of the copies $E_j$ composes the structure $S_1[E_j]$. The defined realizing structure should be tested by the execution of simulation $Z(D_1, S_1[E_j])$ which results in $Q_1$. Comparing the results $Q_1$ with $P_1$ from the definition tests of $T_1$ the designer decides whether the refinement was correct or not. Should be found any contradiction between them, the designer should modify or redesign the structure $S_1[E_j]$ and repeat the simulation. If the comparison satisfies the designer, a successful refinement step has been finished. The remaining task for him is to repeat the above procedure recursively with respect to each type $T_i$ until he achieves a certain stage where all the types used in the lowest level have their realization in the CARS library. This way CARS supports a top-down approach of the design process.

Only two neighbouring levels are taking part in the main design step: the higher level is represented by a type $T_p$ and its realizing structure $S_p$, while the copies $E_q$ used in this structure are copies of the lower level types $T_r$ which are defined by their behavioral descriptions $F_r$.

However, there are features in CARS supporting a bottom-up approach too. It is possible to start with a known realization $R_a$ and compose for it a higher level behavioral description $F_a$ defining the type $T_a$. Using appropriate input data $D_a$ in the simulations $Z(D_a;R_a)$ and $Z(D_a,T_a)$ make it possible to control behavioral equivalence of R and T. This way in further design steps the designer can deal with the behavioral description $F_a$ of type $T_a$ instead of the fine structure of the realization $R_a$ and doing so he can speed up the design process, avoiding data explosion.

## 4. Testing

The designer sits at his terminal and is designing the structure $S_1$ of FIG.1. He decided to realize copies $E_2$, $E_3$, $E_5$, by type $T_3$. Since the behavioral description $F_3$ has not been found in the CARS library, he creates a new type for this purpose.

Translation of the new behavioral description verifies not only synthactical correctness, but ensures semantical completeness and consistency as well. Now he wants to check the functioning of type $T_3$. For this purpose he composes the input data stream $D_3$ and executes the simulation $Z(D_3,T_3)$ and gets the results $P_3$.

Let us suppose the simulation results are satisfactory. The new item in CARS library is the new type $T_3$ defined by the behavioral description $F_3$ together with its definition tests $(D_3;P_3)$. Similarly, he creates the other types needed to realize structure $S_1$ and turns to the definition of the structure itself. The translation and composition of the new structure verifies the completeness and consistency of type $T_1$ and structure $S_1$, so he should compare the functioning of the type and its realizing structure. Executing simulation

$Z\left(D_1, S_1\left[E_j\right]\right)$ gives him the results $Q_1$ which should be compared with the results $P_1$ from the stored tests of $T_1$. If this comparison satisfies the designer the design step is done. Here comes a very important service of CARS the so called test extension. In our example three copies of type $T_3$ were working during the previous simulation namely $E_2, E_3$ and $E_5$. Their transfer data has been kept on the simulation history file under the names $G_2$, $G_3$ and $G_5$. After a succesful simulation CARS automatically extends the stored tests of all types which were referred during the simulation with the data caugth at the I/O points of the copies of the referred types. Thus, for example the stored definition tests $\left(D_3, P_3\right)$ of type $T_3$ will be automatically extended by $G_2, G_3$ and $G_5$, the so called derived tests for $T_3$. Test extension ensures that further refinement of a given type will be tested with respect to all excepted environment.

The tests of the individual types developed quasi-automatically in the synthesis phase of the design this way are to be used as functional tests for the hardware and/or firmware units realizing the same types in the constructional phase.

## 5. Language SHADE/CARS

System CARS is realized by language SHADE /Structured Hardware Design Enforcer/ meeting all requirements defined by the system in chapter 2.

### 5.1. Description of behaviour

In SHADE/CARS the TYPE serves the description of behaviour. The object defined as a TYPE is to be considered as a "black box". It is identified by a name-identifier and a

level identifier. The external connections of this black box: inputs, outputs and/or busses, have to be declared this forming the parameters of the given type.

Type declaration consists of type-statement /heading/ and type body. Form of type statement:

TYPE: ⟨type name⟩(⟨parameters⟩),LEVEL:⟨level identifier⟩.

Type name and the parameters have to be simple identifiers, i.e. strings beginning with letter and containing letters and numbers, the level identifier can be number or simple identifier.

TYPE: NANDGATE(INA, INB, OUTY), LEVEL: ∅.

The body consists of the declaration of the external connections and the definition of the algorithm describing the behaviour of the type.

The external connections are signals, grouped as inputs, outputs and busses. These are described by identifier, width (expressed in bits) and representation (of the signal values).

Designer may specify limitation to set of values used. There is a possibility to define realization of outputs and busses /like open-collector or tristate/ and to indicate the direction of busses. In general, the declaration part of a type

INPUTS:⟨signal declarations⟩OUTPUTS⟨signal declarations⟩.
BUSSES:⟨signal declarations⟩.

⟨signal declarations⟩ is a list of signal declarations separated by semicolons. All signals, having the same attributes can be declared together listing their names, separated by commas and putting the common attributes after the names. Accordingly, form of a general signal declaration:

⟨signal names⟩⟨width⟩ BITS⟨digit number⟩⟨representation⟩
DIGITS,⟨output type⟩,⟨bus direction⟩,
EXCEPT(⟨exception list⟩)

An ⟨output type⟩ can be: NORMAL, OC or TS, direction can be either UNI or BIDIRECTIONAL, while a representation can be either BINary or OCTal, DECimal or HEXadecimal. Some examples of signal declarations:

A,B,C 12 BITS, 3 DEC DIGITS,EXCEPT(6,9);
DOUT 6 BITS, 2 OCT DIGITS, OC;
DATABUS 8 BITS, 2 HEX DIGITS, BIDIRECTIONAL;

The algorithm definition part of the type is based on the stimuli-response method. Both the stimuli and the response take the form of events, input or output event respectively. An event is a change in the value of a signal. An input event consists of an identifier of the event and the description of the change, that forms the event. The change description consists of a signal identifier and the value taken by the given signal.

Form of input event:
⟨event name⟩:⟨signal identifier⟩CHANGES TO⟨value⟩.

SHADE/CARS allows the reference to subsets of the external connections, by using the ANY or ALL keyword together

with the INPUTS, OUTPUTS or BUSSES signal type declarations. The ⟨value⟩ part of the event has to be a definit value, variables and expressions are not allowed. By omitting the ⟨value⟩ part designer indicates that any change in the value means an input event i.e. a stimuli for the type.

Examples:

```
INPUT_EVENTS:
        CLKRISE: CLOCK CHANGES-TO 1;
        INP-CHG: ANY-INPUTS CHANGE;
        A-BUS-SET: A-BUS CHANGES TO ØFFFF HEX.
```

Output events are responses for the input events changing the value of one or more output signals called effect. An output event takes place if an input event triggers it, and certain time dependent conditions are met. The time of the output event is expressed by the delay from the initiating event. An output event is specified this way:

⟨event name⟩: AT⟨time⟩, IF⟨condition⟩,⟨effect⟩

For example:

```
FLOPSET: AT CLOCKRISE+40NS, IF D=1 FROM CLOCKRISE - 10 NS
         TO CLOCKRISE+5NS, Q=1.
```

The above example describes the setting of an edge triggered flip-flop. The Q output of the flip-flop takes the value 1 after 40 nsec of the rising edge of the clock if the D input satisfies a 10 nsec setup and a 5 nsec hold time condition. If there are many output events depending on the same condition, this may be declared separetly as a common condition and a simple reference is allowed to it. Common conditions may be nested without any restriction.

An example of a complete type giving a simplified description of the behaviour of the SN7450 and-or-invert gate:

```
TYPE: AND-OR-INVERT (A,B,C,D,Y), LEVEL: 0.
INPUTS: A,B,C,D 1 BITS.
OUTPUTS: Y.
INPUT-EVENTS: INCHG: ANY-INPUT CHANGES.
OUTPUT-EVENTS: OUTCHG: AT INCHG+22NS, Y=NOT
                    (A AND B OR C AND D).
AND-OR-INVERT END.
```

It must be emphasized again that above example and in general TYPES do not say anything of the internal structure of the described object. It is the designer's task to choose the most suitable structure for the realization of the object.

A very similar approach was taken by Noon in [7], but his proposal covers only behavioral description, moreover it does not distinguish input and output events.

Necessity of this distinction in CARS/SHADE and some other features and elements of the types will be discussed later.

## 5.2. Description of structure

Description of structures in SHADE/CARS is called model. A model has a name and a level identifier also. The same rules control the definition of a model, like a type's:

MODEL:⟨model name⟩, LEVEL:⟨level identifier⟩.

In models all external connections must be declared as INPUTS, OUTPUTS or BUSSES. After the signal declaration, model body consits of two lists: the list of elements

constituing the model and the list of connections among the elements. Elements of a model are realized by types, and as more elements may be realized by the same type, the actual realization is regarded as a copy of that particular type. Copies are identified by proper names indicating actual place of use and /optionally/ parameters.

Definition of element list:

ELEMENTS: ⟨type identifier⟩:⟨copy list⟩.

Types listed in this element list are the component types of the given model, and their level is considered lower than the level of the model.

If there are parameters for a given copy in the element list, then these parameters correspond to those in the type definition. The situation is the same as with the formal and actual parameters of macros in assembly languages, type parameters being the formal and element list parameters being the actual ones.

For example:

```
ELEMENTS:
   NAND-GATE: G1(A,B,Y), G2(C,D,X);
   INVERTER: I1(K,L),I2,I3,I4.
```

This model consists of six elements: two dual input NAND gates and four inverters. G1 gate has inputs A and B, output: Y, inputs of G2 are C and D, output: X, and inverter I1 has input K and output L. The inputs and outputs of I2, I3, I4 are not named in the element list. Signal names as formal parameters in the element list may coincide with names of input or output signals or with parameters of other elements in the list. In both cases system automatically

connects all terminals concerned. This is an implicit definition of connections.

The connection list defines all remaining nets in the model. A net definition is a list of identifiers of connected signals separated by concatenation mark: /'-'/. The designer can refer to any subset of a multi-bit wide signal by using subscripted identifers.

Format of connection list:

CONNECTION:⟨list of nets⟩.

where⟨list of nets⟩ is the definition of all nets separated by semicolons. In this explicit case of definition inputs and outputs of the components have to be identified by the qualified identifier:

⟨element name⟩.⟨name of signal in type declaration⟩

Qualified names may be subscripted too.

An example for a model, describing /a possible/ structure of the above defined and-or-invert gate:

```
MODEL: AND-OR-INVERT, LEVEL:1
INPUTS: A,B,C,D.
OUTPUTS: Y.
ELEMENTS: AND-GATE: AG1,AG2;
          NOR-GATE: NG.
CONNECTIONS:
A-AG1.IN1; B-AG1.IN2; C-AG2.IN1; D-AG2.IN2;
AG1.OUT-NG.IN1; AG2.OUT-NG.IN2;NG.OUT-Y.
AND-OR-INVERT END.
```

The same structure can be described by making use of the parameters. The connection list can be omitted if the element list is the following:

ELEMENTS: AND-GATE:AG1(A,B,X1), AG2(C,D,X2);
NOR-GATE: NG(X1,X2,Y).

The structure that follows from this example is on Fig.2. It is obvious that a model can be transformed into an executable simulation program only if definition of all types, having at least one copy in the model is given. On the other hand, the model does not assume anything of the inside realization of composing types, it is independent of the actual level or complexity of types, consequently it is level independent.

5.3. Language elements for refinement and enroughment

The most important concepts of SHADE/CARS with respect to the refinement and enroughment process are the OPERATION and SYNONIM. OPERATION is the means for description of refinement and enroughment in the time domain while SYNONIM serves the same purpose in data structure domain.

General form:

OPERATIONS: ⟨operations⟩

After the OPERATIONS keyword and the: delimiter one or more ⟨operations⟩ may be specified.

Form of one operation:

⟨operation identifier⟩:⟨time and condition⟩;⟨effect part⟩.
⟨operation identifier⟩ END.

The prefix ⟨operation identifier⟩: and the suffix ⟨operation identifier⟩END. flank the description body.

In the ⟨time and condition⟩ part start and stop time can be specified for the operation.

Form of start specification:

STARTS-AT ⟨time expression⟩,⟨condition⟩

Form of stop specification:

TERMINATES-AT ⟨time expression⟩OR⟨condition⟩

If a start condition is specified, operations will be executed only if the condition is met. If stop is specified with condition the execution of the operation specified by the effect part will be ended if this condition is met.

Form of effect part:

⟨normal effect part⟩,⟨early stop part⟩

In the normal effect part there are assignments specifying the value of the signals at stop time. The value of these signals are unavailable during the operating time. When the operation is ended because of the condition in the stop specification, the value of the signals referred in the effect part those specified in the early stop part by assignments.

In the effect part the next statements may be used: simple assignment, IF assignment, ON assignment. On the right side of the simple assignment expressions including arithmetical and logical operators may be specified. The IF assignment consits of one or two simple assignments and a condition. If the condition is met, the first simple assignment

/after the THEN keyword/ will be executed, otherwise the second one /after the ELSE keyword/. Second assignment and keyword ELSE are optional.

In the ON assignment more assignments may be given. Depending on the value of the referred variable the appropriate assignment will be executed.

In the top-down design process not only TYPES are decomposed into STRUCTUREs but OPERATIONs are decomposed into lower level OPERATIONS and/or EVENTs as well. This feature of SHADE/CARS helps the designer to make time refinements.

Nevertheless, the bottom-up approach is also supported by the OPERATION concept since different EVENTs and/or OPERATIONs can be composed into a higher level OPERATION, thus eliminating unimportant details of timing considerations at a certain level.

The SYNONIM concept is used in a similar way in the data representation of SIGNALS. By SYNONIM declarations one can refer to constituents of SIGNALS, or lower level SIGNALs can be composed into a more complex form. For trivial cases such as hexadecimal/binary and vice versa, etc. standard data representation transformers can be used but for more sophisticated cases it must be left for the designer to define his own data transformers.

## 5.4. Formal test of completeness and self-consistence

When describing the behaviour of a digital system the two most common types of errors are: the description contains contradiction and/or there are input combinations with no defined response. It is rather tiresome to detect

these errors by simulation , it is desirable to eliminate them by formal methods, according to basic concept of SHADE/CARS.

By restricting the ⟨value⟩ in the input events only to defined values it became possible to check the existance of values on the inputs not covered by any specified input event. This too is the way to find out if an input event refers to an invalid value according to limitations specified for the given signal. The separation of input and output events makes it easy to test, whether all output events have their triggering input events. But the most important reason of this separation is to grant the possibility of composed event definition. A composed event is a partially ordered set of events. This is a special application of the concept of path expression [S], connected with a time expression. This new element of the language expands the possibilities for refinement/euroughment. The designer can describe sequences of events as a composed event and refer to it later only by its identifier.

The composed event may be: sequence, selection, repetition, set, list and member.

In case of a sequence, order of composing events is defined and the time elapsed between two consecutive events is specified either by a defined time value, or a time interval defined by minimal and maximal values or can be unrestricted. The form of a sequence:

⟨event name⟩: SEQ-OF: ⟨event name⟩, ⟨event name⟩
⟨time interval⟩, ... ⟨event name⟩⟨time interval⟩

For example, the event:

COMPEV: SEQ-OF: A,B(1Ø), C(∗/2Ø), D(3Ø/6Ø), E(1Ø/∗), F means that the A,B,C,D,E,F events have to take place in this order, and between A and B a delay of 10, between B and C at most 20, between C and D at least 30 but at most 60, between D and E at least 10, and between E and F an undefined number of time units is allowed. The unit of time delay can be specified, default unit is nanosec.

The selection does not establish any order in sequence for the composing events. They can happen in any sequence, but all of them have to take place. The designer can set up conditions for the time between events. The form of a selection:

⟨event name⟩ : SEL-OF: ⟨event name list⟩AT-INTERVALS-OF (⟨time interval list⟩)

In repetition the number of occurance of the composing event is prescribed with a given timing. The form of repetition:

⟨event name⟩ : REP-OF: ⟨event name⟩⟨time interval⟩,⟨cycle⟩ TIMES

Event types: set, list and member define special subsets of the listed component events. The set defines any possible subset, the list requires that all components should happen simultaneously, while the member means that only one of them can occur. The optional time parameter means a deadtime, i.e. repeated occurance of the component events within this interval will be ignored.

Some examples of composed events:

SELEV: SEL-OF:A,B,C,D AT-INTERVALS-OF(5,10,π/40);
REPEV: REP-OF:A(10/25)3 TIMES;
SETEV: ANY-OF:A,B,C,D,E(10NS);
LISTEV: ALL-OF:A,B,C(10NSEC/1USEC);
MEMBEV:ONE-OF:E,F,G,H(π/100).

Component events themselves can also be composed events and they can be nested in any depth.

The composed events are not only easy to use tools for refinement and enroughment, but they open up a new field for the formal testing methods, too.

Let us consider two descriptions of the same digital system. One is the behavioral description, a type, the other is a structural one, a model. Let us assume that all the elements, appearing in the model are defined, thus both type and model can be tested by simulation, using the same input data.

The result of the two tests must be identical if the model is correct. This comparison is the last step of the process. Test set is specified by type descritption as it must contain all functions expected from the given type. The model on the other hand may have additional, unspecified functions too. This way a positive result of the comparison don't mean identity of type and model, but correspondence indicating only that model fulfils specified requirements.

If the same model is tested for a different type description, as necessarry in bottom-up approaches, it may or may not fulfil additional or changed requirements.

Comparison means an evaluation, whether the set of events describing the behaviour of the type is a complete subset of possible events of the respective model or not.

This evaluation may be done at different levels. It is obvious that on lowest level the evaluation is easy, because only numeric result-values of simulation must be compared, but simulation must be done for all possible values and combinations of inputs withint the region of specified input value sets.

Investigations are continued to solve evaluation on higher levels, i.e. corresponding input and output events of type description should be applied to respective model directly. It must be still proved that applied methods for formal evaluation of correspondence are correct in the nontrivial cases too.

## 6. Implementation of SHADE/CARS

The compiler for SHADE/CARS is written on CDL2 [9]. It consists of approximately 2000 rules, runs on an IBM 370/115 and a SIEMENS 7755. The compiler performs complete syntax analyzis, and checks the types for completeness and types and models for self-consistence. From the source text it compiles an assembly language source program. This program consists of merely calls of predefined macro instructions. These macros, a dynamic simulator and an output program forms a runtime system for the programs generated by SHADE/CARS compiler.

Summary

System CARS has been developed to assist the designer in the synthesis phase of the design of digital systems. CARS applies a top-down structured approach in the design process and can be used at any user defined level of refinement. CARS does not apply any restriction on the creative work of the designer, but it only gives means to describe his concepts and tools to check his results. The design language defined for CARS is well suited to describe behavioral concepts and structures as well. One of the most important feature is, that the designer can use the same SIMADE/CARS language (for both purposes), CARS supports the consistency analysis of the refinement level achieved and it not only compares neighbouring levels of the same design, but at the same time it derives functional tests for the lower level structure. At the end of the design process these tests are automatically composed into a functional test of the hardware designed. Each level of the design is kept in the CARS library and can be modified at any moment of the design process. When modification takes place CARS makes a thorough analysis of it thus enforcing correctness. When using multilevel simulation as an analysis tool of CARS, the system provides standard means for data and time refinements and enroughments as well.

[1] Bohus M., Csopaki Gy., Filp A.: Simulation of Computers and their Components. Techn. Report for the ICCT Budapest 1979. /in Hungrasion/

[2] Máté,L., Bohus,M., Filp,A., Hinsenkamp,A.: CARS: A Computer Aid for Recursive Synthesis. Proc. of ICCC'80.

[3] Coplaner H.D. and Janku,J.A.: Top Down Approach to LSI System Design. Computer Design vol. 13. No.8. Aug.1974. pp. 143-148.

[4] Hinsenkamp,A.: Simulation as a Computer Aid for Hardware Dosign. Tech.Memo ICCT Hardware Lab. Budapest 1978. /in Hugarian/

[5] Hill,D., van Cleemput,W.: SABLE: a Tool for Generating Structured Multi-level Simulation. Proc. of 16th Design Automation Conference San Diego 1979. pp. 272-279.

[6] Máté,L.: A Program Package for Multilevel Simulation of Digital Systems. Tech Project CAI.HAS, Budapest 1976. /in Hugarian/

[7] Noon,W.A.: A Design Verification and Logic Validation System. Proc. of the 14th Design Automation Conference New York 1977.

[8] Compbell,R.H., Habermann,A.N.: Process synchronization by path expressions. Lecture Notes in Computer Science No.16. New York 1974.

[9] C.H.A.Koster: CDL-A Compilor Implementation Language Springer Verlag, Berlin 1977.

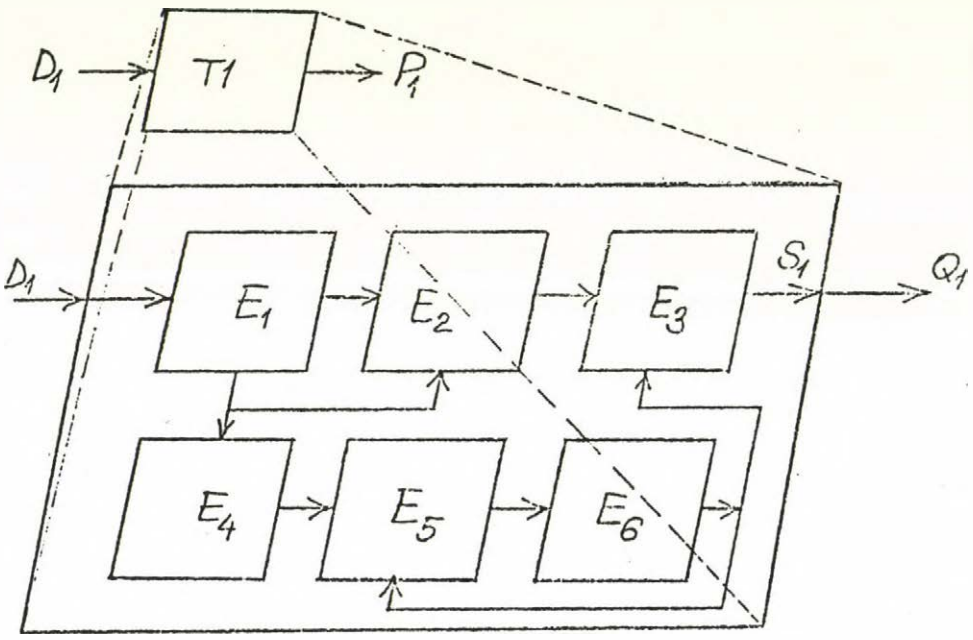
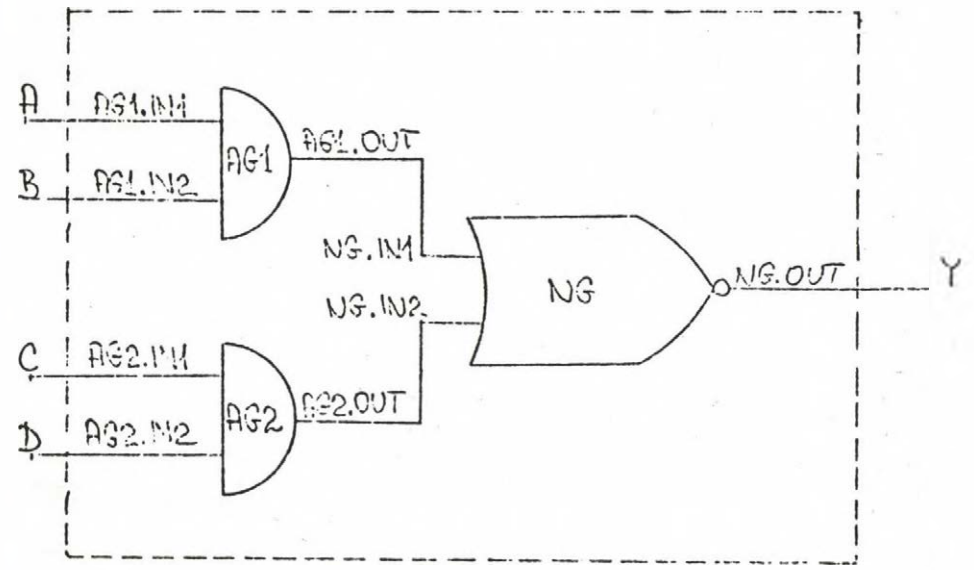



Fig. 1. The design step



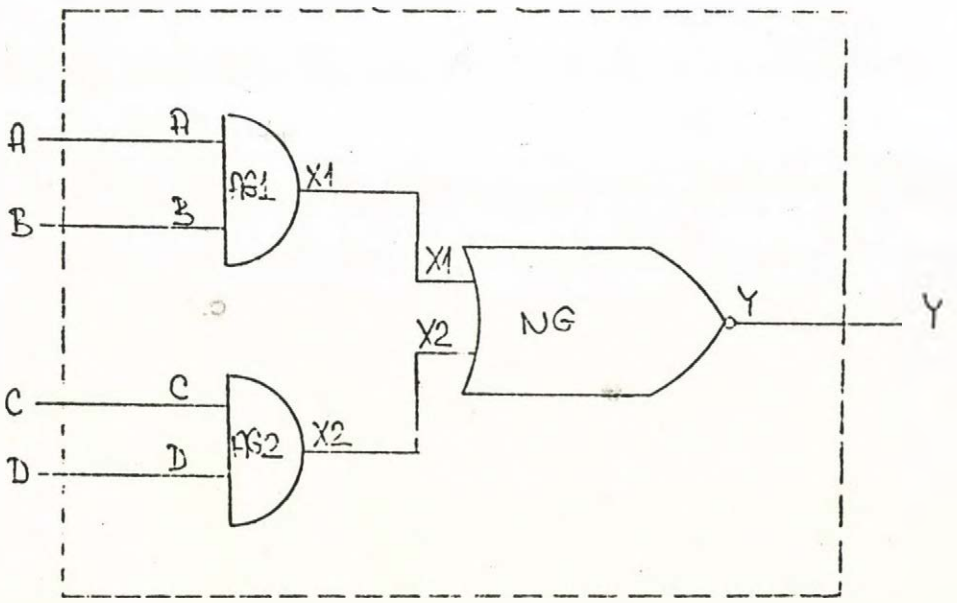




Fig.2 The structure of the model

MTA SZTAKI Tanulmányok 147/1983 229-231

# TEACHING AND LEARNING THE HIGHWAY CODE THROUGH PERSONAL COMPUTERS

R. Mitkov

Institute of Mathematics with Computer Centre
Bulgarian Academy of Sciences

The rapid development of microcomputer industry offers through its latest products new possibilities. The personal computers nowadays are comparatively cheap, possess satisfactory quantity of memory and can be programmed at least in BASIC. What is more, their portability and acessibility make them comfortable for daily popular use.

It is clear, that the personal computers can be used in much more important and interesting fields, than the family income and outcome calculation. The personal computers are very suitable for and will probably be soon the basic instrument in the computer-aided instruction. The personal computer enables the subject teaching both by text and picture and sometimes even by sound. It can not only teach students, but also check them.

The essential role, which the personal computers play and will play in the education attracts various state officials in many industrial countries. In U.S.A. there was an official proposal presented to the Congress concerning the foundation of "National center for personal computers in education" / 1 /. In France the large organisation AFPA, which is responsible for the "professional education of adults", carries out all its instruction by means of personal computers /3/. The results of many schools, which experiment computer-aided instruction by personal computers turn out to be very promising /1/, /2/.

The laboratory of mathematical linguistics at the institute of mathematics with computer centre /part of the Bulgarian

Academy of Sciences/ has focused its attention on the personal computers as an important tool in the computer-aided instruction. Along with the traditional educational fields, an interesting experiment was carried out to computerize the teaching and learning /and checking the obtained knowledge/ the highway code. Personal computers Apple II were used, which provide also excellent colour graphics capabilities.

The highway code in Bulgaria is devided into chapters and the exam, organized by traffic police personel, represents certain tests, each test element of which contains three preliminarily given answers and only one of those is correct. The test has purely textual questions, as well as questions, related to pictures, representing traffic signs or traffic situation.

The subject, divided into chapters was displayed on the screen of Apple II. After reading and learning each chapter the student is controlled if he has mastered it. If not, he goes once again through the whole chapter. The colour graphics of Apple II provides 16 different colours in usual graphics and 8 colours in high resolution graphics. Both kinds of graphics were used to draw nicely various road signs and traffic situations.

Except for teaching the highway code, the personal computers were used to check the knowledge of each student, i.d. computerizing the traffic police exam. Every student receives a test and after giving his answers the computer "decides" whether he has suceeded or failed. The tests are not only textual, they contain both text and picture. Practically, unrestricted number of different /but of equal difficulty/ tests can be generated by the personal computer.

In general, we can distinguish three levels in teaching the highway code through personal computers:

1. Text - the personal computer is used only to "type" the text on its screen.

2. Text + static picture - besides the text there are colour pictures, representing road signs and traffic situations.

3. Text + dynamic picture - moving objects /vehicles and pedestrians/ can be generated on the screen. These can be used as computerized games, using traffic rules or to represent a certain traffic problem to be solved.

The programs are written in BASIC Applesoft, which contains additional operators in relation with the usual and high resolution graphics of the personal computer Apple II.

References

1. Th. Esbensen - "Elevated education made easy: computers in schools" - Personal computing/October '81

2. Th. Esbensen - "Personal computers: the golden mean in education" - Personal computing /November '81

3. Entretien: "Des ordinateurs individuels - pour la formation professionel des adultes" - L'Ordinateur Individuel /Octobre 1980